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

Mario Montagud[1], Fernando Boronat[1], Hans Stokking[2], and Pablo Cesar[3]
[1]Universitat Politècnica de València (UPV)
Grao de Gandia, Valencia (Spain)
mamontor@posgrado.upv.es; fboronat@dcom.upv.es
[2]TNO: Netherlands Organisation for Applied Scientific Research
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 users are gaining momentum. In these networked scenarios, synchronized playout between the involved participants must be provided to enable truly interactive and coherent shared media experiences. This research topic is known as Inter-Destination Media Synchronization (IDMS). This paper presents the design and development of an advanced IDMS solution, which is based on extending the capabilities of RTP/RTCP standard protocols. Particularly, novel RTCP extensions, in combination with several control algorithms and adjustment techniques, have been specified to enable an adaptive, highly accurate and standard compliant IDMS solution. Moreover, as different control or architectural schemes for IDMS exist, and each one is best suited for specific use cases, the IDMS solution has been extended to be able to adopt each one of them. Simulation results prove the satisfactory responsiveness of our IDMS solution in a small scale scenario, as well as its consistent behavior, when using each one of the deployed architectural schemes.

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

List of Abbreviations:

AMP	Adaptive Media Playout
APP	(RTCP) Application-Defined Packet
CBR	Constant Bit Rate
DCS	Distributed Control Scheme
ETSI	European Telecommunications Standards Institute
GoP	Group of Pictures
GPS	Global Positioning System
IDMS	Inter-Destination Multimedia Synchronization
IETF	Internet Engineering Task Force
IPTV	Internet Protocol Television
FTP	File Transfer Protocol
M/S	Master/Slave
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
RTT	Round Trip Time
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
XR	(RTCP) Extended Report
WebRTC	Web Real-time Communication

1. Introduction.

Research on media synchronization (sync, hereafter) has been primarily focused on orchestrating the playout of incoming media streams locally at single devices. On one hand, intra-stream sync deals with the maintenance of the temporal relationships between the Media Units (MUs), e.g. video frames or audio samples, of a specific media stream. On the other hand, inter-stream sync refers to the preservation of the temporal dependences between MUs of independent, but correlated, media streams, being lip-sync the most representative use case [1, 2]. Both types of media sync techniques are essential in distributed media applications.

Recently, novel forms of media sharing applications are gaining momentum [3], allowing geographically distributed users to socially interact (e.g., using text, audio or video chat) within the context of simultaneous content consumption. Some examples include Social TV, synchronous e-learning, and networked multi-player games. However, many challenges must be faced to enable truly interactive and coherent shared media experiences [4]. One of the major technological barriers is the lack of mechanisms to guarantee a simultaneous sync of the media playout across the involved devices, which is known as Inter-Destination Media Sync (IDMS) [3, 5, 6]. IDMS is needed to adaptively compensate the end-to-end (e2e) delay variability that is originated when delivering specific media content to a group of distributed devices. Indeed, previous studies have revealed that the magnitudes of these (playout) delay differences are significantly larger than acceptable limits in most of the IDMS use cases (e.g., [3, 7]), leading to a severe QoE (Quality of Experience) degradation [8, 9].

In [1, 10], the authors presented the design, implementation and evaluation of a preliminary version of a centralized IDMS solution, which is based on simple extensions to the RTP/RTCP standard protocols [11]. Experimental tests proved its satisfactory performance in both real, but controlled, scenarios [1] and simulated ones [10]. However, with the advent of distributed media consumption, further requirements emerge. First, inter-operability between (third-party) implementations and devices needs to be guaranteed. Second, some IDMS use cases require very strict sync levels [3], so highly accurate IDMS solutions must be provided. Furthermore, our study in [3] pointed out that a centralized approach is not always the best choice for IDMS, as in some cases distributed solutions are more suitable.

The main goal of this work is to meet the above requirements. Accordingly, the contribution of this paper is three-fold. First, novel standard compliant RTCP extensions[1] for IDMS have been designed to guarantee compatibility and high accuracy. Second, our RTP/RTCP-based IDMS solution has been extended to be able to adopt different control or architectural schemes. This allows our IDMS solution to be adaptive and flexible enough to be efficiently deployed in different networked environments. Likewise, newer aspects and functionalities have been added to improve

[1] Both ETSI [12] and IETF [13] standardization bodies have adopted our RTP/RTCP-based technology for IDMS.

its performance, for each scheme in use. Third, our RTP/RTCP-based IDMS solution has been implemented in a simulation framework and its consistent behavior and satisfactory responsiveness are proved through simulation tests in a small scale setup. Further research will be targeted to analyze the scalability and adaptability of our IDMS solution by performing a thorough evaluation in large scale and dynamic settings.

The structure of the paper is as follows. In the next section, the state-of-the-art regarding IDMS is reviewed. In Section 3, we describe the main components of our IDMS solution, for each control scheme. Section 4 gives performance results. Finally, Section 5 provides our conclusions and a discussion of future work.

2. Related Work

Due to the increasing relevancy of IDMS, several works have addressed this topic. In [5], authors presented a survey of IDMS solutions, whilst in [3] we presented a compilation of IDMS use cases. Besides, the recent work in [6] provides a historical review of multimedia sync techniques (including IDMS).

Two main categories of IDMS solutions can be distinguished (see [5, 6]): *axis-based*, which aim to continuously align the presentation of media streams along either a virtual or a wall-clock timeline axis; and *control-based* (a.k.a. *event-based*), which handle the sync of media streams over a discrete set of reference control points or events.

In general, existing IDMS solutions were designed for specific applications, such as audio/video streaming (e.g., [1], [2], [9], [14]), conferencing (e.g., [15-18]), gaming (e.g., [19-22]), collaborative virtual environments (e.g., [23], [24]) or synchronous e-learning (e.g., [15], [17]). Likewise, these solutions involve different types of media streams, such as audio, video, haptic media (e.g., [16], [17]), and user-generated events (e.g., [4], [20], [21]). It is important to note that most of them are ad-hoc solutions, rely on the definition of new (application-specific) proprietary protocols, and make inter-operability between implementations and domains very difficult.

In addition, as such solutions were devised for specific networked environments and for meeting specific requirements, they differ on the adopted architectural approach to exchange the required information about IDMS. In this context, three main IDMS control schemes have been identified (see Fig. 1): two centralized (Master/Slave, or M/S, Scheme and Sync Maestro Scheme or SMS) and one distributed (Distributed Control Scheme or DCS). Readers are referred to [3] to find an exhaustive description, classification (summarized in Table 1) and a qualitative comparison among these IDMS schemes. Even though it was concluded in that work that SMS is, in general, the best-ranked scheme for IDMS, the appropriateness of DCS and M/S schemes for specific use cases was also identified. 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.

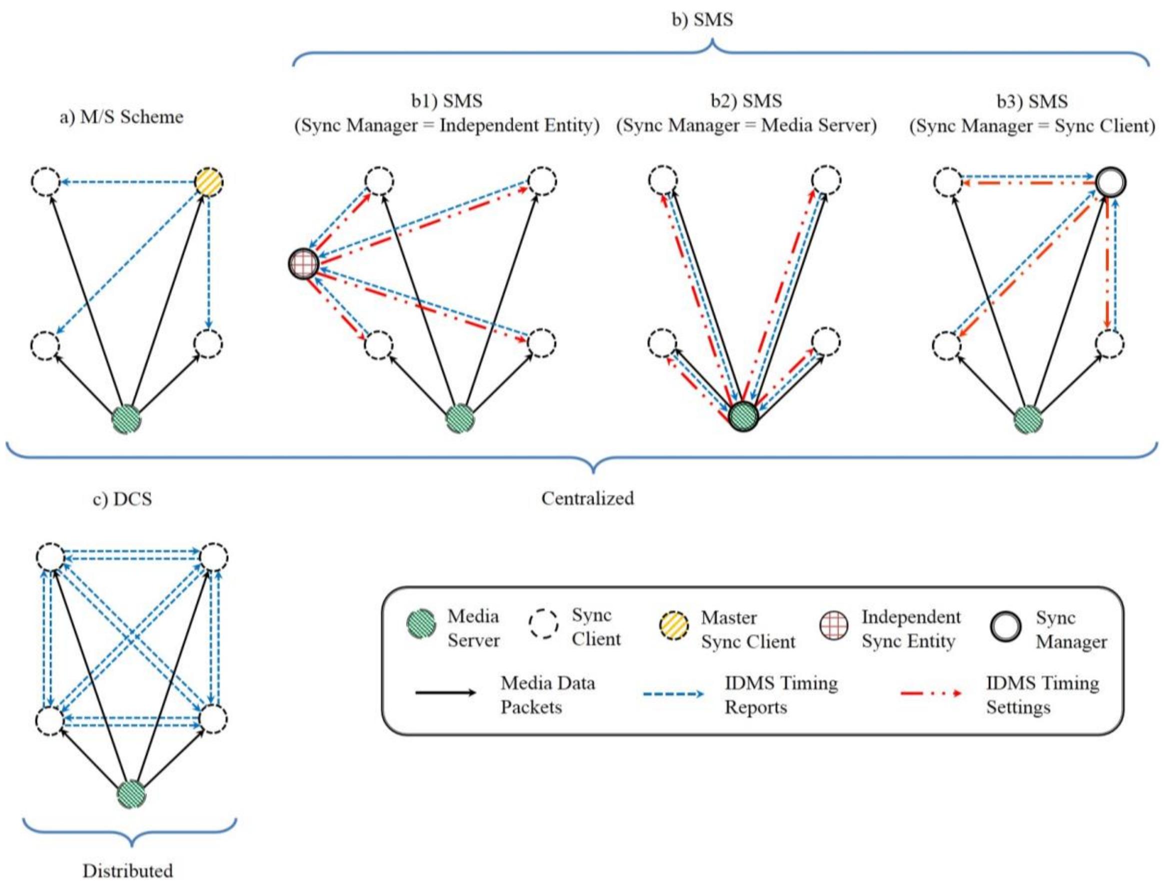

Fig. 1. Control Schemes for IDMS.

Table 1 Classification of IDMS Solutions based on the Adopted Control Scheme

	CENTRALIZED		**DISTRIBUTED**
	M/S Scheme	**SMS**	**DCS**
IDMS Solutions	[4], [9], [14]	[15], [23], [16], [17], [5], [10]	[20], [21], [22], [17], [4]

In general, the specification of an IDMS solution is highly dependent on the context and environment for which it is going to be deployed. For instance, some applications can be small-scale, while others can be deployed over large-scale settings. Some others require the achievement of stringent sync levels, while lower sync accuracy may be acceptable in other solutions. Bandwidth availability and multicast feedback capabilities can be an issue (or not) in specific scenarios. Likewise, other aspects, such as delay minimization or robustness, can be especially relevant in particular environments. At the end, the targeted use cases dictate the requirements, which will determine the necessary characteristics and functionalities of the media sync solution under design.

Our intention was to design a highly accurate, wider applicable (i.e., easy to deploy and valid for different use cases) and inter-operable IDMS solution, by using standard compliant components (unlike other solutions). Accordingly, after considering several candidate protocols for being extended for IDMS purposes, we chose RTP/RTCP as the core protocols for designing our IDMS solution, mainly because of their compliance with our targeted requirements [25]. A thorough discussion of the convenience and advantages of using and extending RTP/RTCP for IDMS purposes can be found in [25]. Likewise, that paper claims the relevancy of such protocols in current and future deployments, both in managed streaming environments, such as IPTV, and in multi-party browser-to-browser WebRTC scenarios [26, 27].

In [1, 10], we presented a preliminary version of our RTP/RTCP-based IDMS solution, which was based on the use of SMS. Our IDMS solution is mainly *axis-based* (by using the timelines provided in RTP packets), but it also relies on the exchange of regular RTCP packets to monitor and control the overall sync process (*control-based*). The proposed RTCP extensions were originally targeted for our own-designed (vendor-specific) applications (e.g., video sharing or surveillance), while: i) adding a minimal traffic overhead; and ii) operating within controlled scenarios. However, given the promising results that were obtained in those works and the emerging requirements for enabling inter-operability and high accuracy, the need for an evolved and standardized IDMS solution arose. Moreover, because of the suitability of the different control schemes for specific use cases [3], we also decided on the deployment of DCS and M/S schemes. This way, our IDMS solution can be efficiently deployed in different scenarios, according to specific requirements or available resources.

3. Standardized RTP/RTCP-based IDMS Solution

This section describes all the components of our RTP/RTCP-based IDMS solution and discusses the improvements that have been added. To aid understanding this paper, Table 2 lists the symbols defined in this section and their meaning.

Table 2 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

3.1. *Definition of new RTCP messages for IDMS*

We have been working on specifying a standardized RTP/RTCP-based technology for IDMS within the umbrella of ETSI TISPAN [12] and IETF [13]. In particular, two new RTCP messages have been defined [13], following the guidelines specified in RFC 5968 [28]. First, an RTCP XR block for IDMS, called *IDMS report*, has been specified to enable Sync Clients to provide feedback about reception and/or presentation times for specific RTP packets (see Fig. 1). Concretely, this IDMS report contains the following metadata for IDMS: i) the Payload Type (PT) of the RTP stream this block reports on; ii) the SSRC of the source of the RTP stream this block reports on; iii) the Sync Group Identifier to which the sender of this report belongs; iv) the generation 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). Second, a new RTCP packet type for IDMS [13], called *IDMS Settings packet*, has been defined to allow the Sync Manager (in case of using SMS, see Fig. 1) to provide a common target playout point to which all the Sync Clients belonging to a specific group must match. Additionally, a new Session Description Protocol (SDP) parameter, called *rtcp-idms*, has been specified [13] to inform about 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). The use of this SDP attribute for groups' establishment and management is discussed in [13].

Apart from guaranteeing inter-operability, the proposed (standard compliant) extensions in our IDMS solution provide significant improvements in terms of flexibility, scalability and accuracy compared to our preliminary version [1, 10]:

1) Our preliminary IDMS solution was based on the use of NTP (Network Time Protocol) for wall-clock sync. In this evolved version, we take advantage of the SDP attribute defined in [29] to enable the negotiation and selection of a common (or related) wall-clock source by all the involved sync entities. Apart from NTP, other (more accurate) technologies for clock sync, such as PTP (Precise Time Protocol) and GPS (Global Positioning System), are supported.
2) The newly defined IDMS report allows reporting on both packet reception and presentation times. The first option is easier, but the downside is that the delay variability that is originated at the client side (e.g., due to buffer settings, decoding, processing and rendering delays) is not taken into account. This can incur in a loss of accuracy. The second option enables highly accurate e2e sync, but requires tracking RTP packets until their ultimate presentation times (application layer). This feature could not be provided in some RTP implementations, as discussed in [13]. Therefore, reporting on packet arrival times is mandatory, but reporting on presentation times is optional in the evolved version of our IDMS solution. In our simulation-based prototype, the second option is supported by all the involved sync entities. Therefore, syncing on RTP arrival times has not been considered in this paper.

3) The RTCP IDMS Settings packet includes a 64-bit presentation timestamp for indicating the target presentation times. This allows a high level of granularity for those use cases presenting stringent sync requirements [3, 13].
4) The new RTCP messages use a 32-bit field for allocating the group identifier, instead of an 8-bit field used in [1, 10]. This minimizes the probability of collision (i.e., two groups choose the same identifier) and simplifies the processes of generating random identifiers [11].
5) The new RTCP messages use RTP generation timestamps for identifying specific RTP packets, instead of RTP sequence numbers used in [1, 10]. This helps solving fragmentation issues when application-layer MUs (e.g., video frames) are carried in several RTP packets, since all of them will include the same timestamp. Moreover, a major advantage is that it allows inferring the asynchrony between the involved Sync Clients, even when using Variable Bit Rate (VBR) coding mechanisms (but with constant MU rate, e.g. a Media Server transmitting 25 fps). This could not be assured when using RTP sequence numbers, because their rate of advancement depend on the temporal and spatial compression methods employed in those VBR coding mechanisms, as well as on its configured parameters (e.g., Group of Pictures or GOP size and pattern, quantizer scale, etc.), thus being only applicable for Constant Bit Rate (CBR) traffic patterns.

3.2. *Exchange of IDMS Messages in each Control Scheme.*

During an RTP session, each distributed *i-th* Sync Client regularly sends RTCP Receiver Reports (RR) to inform about QoS (Quality of Service) metrics [11]. Additionally, each *i-th* Sync Client must send an IDMS report in each compound RTCP packet, 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 when using SMS or multicast when using DCS) or only by the master Sync Client (multicast when using 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 session.

This way, once the overall information about IDMS 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 (or e2e) 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[2] 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):

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

$$\Delta^k_{\max} = (d^k_{\max} - d^k_{\min}) \tag{3}$$

If 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.

3.3. *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 selecting the 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 [10]. Possible strategies include: sync to the slowest Sync Client; sync to the fastest Sync Client; sync to the mean playout point; and sync to the nominal rate of the Media Server. Additionally, 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 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 this new version, the processes of comparing the IDMS timing of the distributed Sync Clients (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 preliminary IDMS solution in [1, 10], 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

[2] 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 Sender Reports (SRs) sent by the Media Server [11].

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 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 IDMS, 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 media session advances in time. Hence, additional adaptive buffering control techniques should also be included. This policy could not be applied in live streams, because of the inability to speed up a live stream.

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 [5].

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 other 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 unpredictable. As in the above policies, adaptive buffering control techniques should be deployed in future studies.

In all the previous 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 sync 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 reasonably stable, underflow and overflow situations will be avoided. This is because the playout states of the deviated Sync Clients in each group will 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.

3.4. *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 asynchrony with the selected IDMS reference, are described. Both processes have been also enhanced in this evolved version of our IDMS solution.

When using SMS, if an 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 [11]. 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)[3] and by a given MU sequence number identifier (MU_i^k), at p_i^k instant (local playout point). That Sync Client would consume the successive MUs with a (possibly deviated)[4] 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[5]) 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:

[3] We are assuming that each RTP packet encapsulating (a portion of) the same MU will include the same (or very close) RTP timestamp.
[4] The playout rate deviations and their effect on media sync are explained in [10].
[5] We are also assuming here that the MUs are captured/generated periodically, e.g. a constant transmission MU rate of $\theta \approx 25$ MU/s. Therefore, we can identify which MU 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 RTP media stream, and the RTP and NTP Timestamps can be mapped according to the info included in each RTCP SR [4].

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

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 ($d^k_{n,i}$):

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

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 control scheme in use, the Sync Clients must adjust their playout processes to achieve IDMS. It can be done by following two possible reactive techniques. 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 it is similar to the one when using DCS, but in such a case only the master Sync Client multicasts IDMS reports.

3.5. *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 drastically affected in any of the IDMS 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 [11].

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 in order to collect the overall IDMS timing. So, a control timer has been included to manage the triggering of necessary playout adjustments. Accordingly, the playout adjustments can be triggered either as a result of the detection of an out-of-sync situation 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 IDMS 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 when using M/S Scheme. 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.

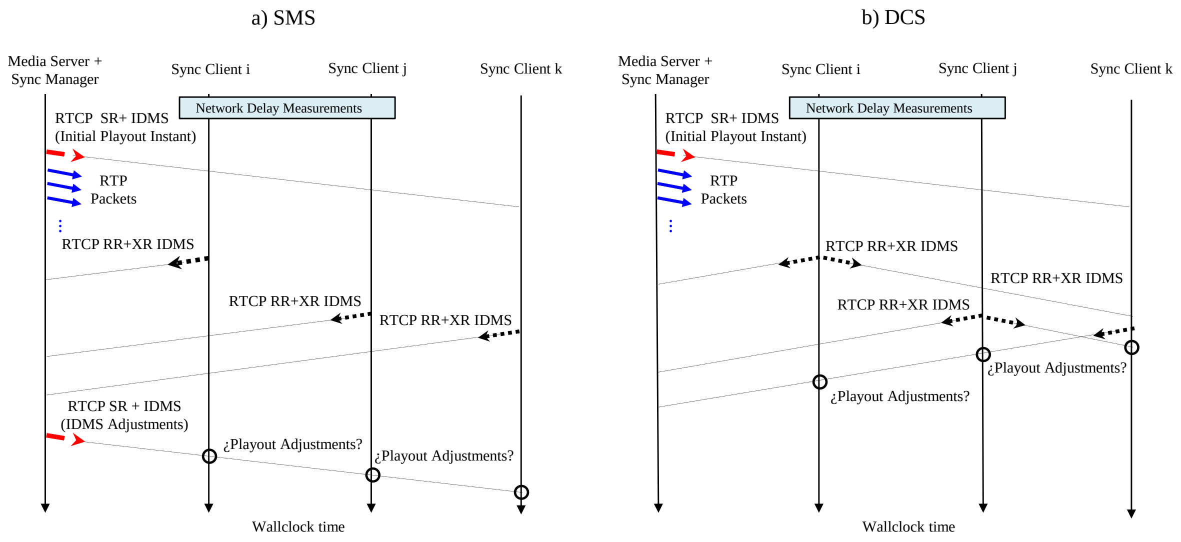
a) SMS
b) DCS
Media Server + Sync Manager
Sync Client i
Sync Client j
Sync Client k
Network Delay Measurements
RTCP SR+ IDMS (Initial Playout Instant)
RTP Packets
RTCP RR+XR IDMS
RTCP SR + IDMS (IDMS Adjustments)
¿Playout Adjustments?
Wallclock time

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

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.

3.6. *Playout Adjustment Techniques*

3.6.1. *Aggressive 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)[6]. 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 presentation time for one MU[7] (Fig. 4), thus probably causing a noticeable playout discontinuity/disruption.

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

3.6.2. *Smooth Adjustments: Adaptive Media Playout (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

[6] The modeling of the playout rate deviations (skew -γ- and drift -ε-) and their effect over the media sync are described in [10]
[7] We are assuming that only entire MUs can be skipped using our developed aggressive adjustment policy.

(e.g., [30-32]). 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 (e.g., [30, 31]) and, occasionally, the inter-stream sync quality (e.g., [32]). However, our work in [10] proposed to extend the use of AMP for IDMS purposes. By using AMP, distributed Sync Clients were able to smoothly acquire an overall sync 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 operate with VBR traffic patterns, as well as adapted to be applied in each one of the deployed IDMS 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. 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 scenario.

The operation of our 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.

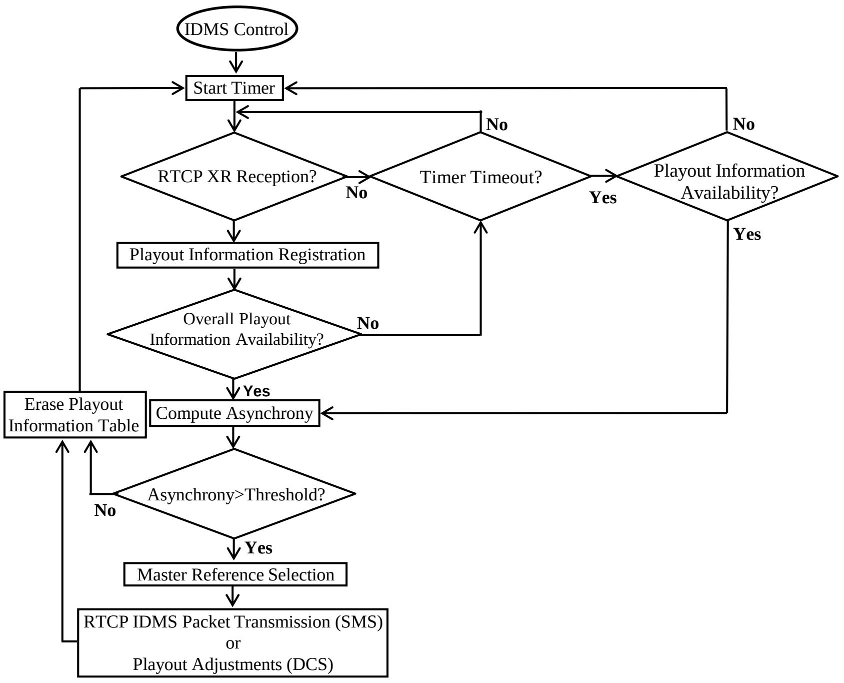
IDMS Control
Start Timer
RTCP XR Reception?
No
Timer Timeout?
No
Yes
Playout Information Availability?
No
Yes
Playout Information Registration
Overall Playout Information Availability?
No
Yes
Erase Playout Information Table
Compute Asynchrony
Asynchrony>Threshold?
No
Yes
Master Reference Selection
RTCP IDMS Packet Transmission (SMS) or Playout Adjustments (DCS)

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

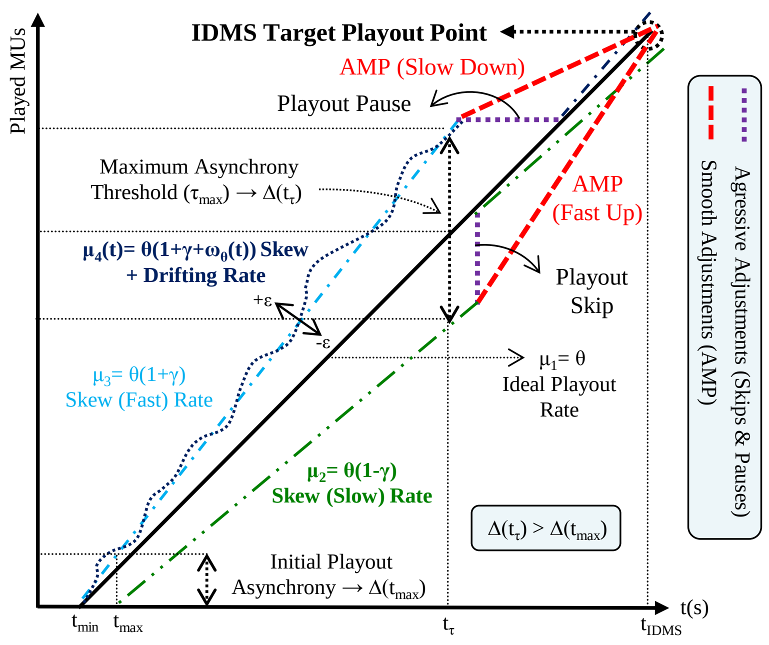
IDMS Target Playout Point
AMP (Slow Down)
Playout Pause
Maximum Asynchrony Threshold (τ_{max}) → $\Delta(t_\tau)$
$\mu_4(t)= \theta(1+\gamma+\omega_0(t))$ Skew + Drifting Rate
+ε
-ε
$\mu_3= \theta(1+\gamma)$ Skew (Fast) Rate
AMP (Fast Up)
Playout Skip
$\mu_1= \theta$ Ideal Playout Rate
$\mu_2= \theta(1-\gamma)$ Skew (Slow) Rate
$\Delta(t_\tau) > \Delta(t_{max})$
Initial Playout Asynchrony → $\Delta(t_{max})$
Played MUs
t(s)
t_{min}
t_{max}
t_τ
t_{IDMS}
Agressive Adjustments (Skips & Pauses)
Smooth Adjustments (AMP)

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

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 (speed up) or decrease (slow down) the video playout rate in order to minimize the detected asynchrony with the master, $\Delta^{k}_{n,i}$ (see Eqs. 4 and 5). This can be accomplished by means of increasing/decreasing the presentation period of all remaining MUs to reach the IDMS target playout point 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 version of our 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).

A key factor to perform AMP is to determine the allowed ratio within which the video playout speed can be varied without being annoying to the users' perception. Previous subjective studies have shown that playout speed variations of up to 25% are often *unnoticeable* to users [30, 31]. We rely on such assumptions and define a 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 value of this parameter is computed to get playout adjustments 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 must be set.

The flow chart of the AMP algorithm using SMS is sketched in Fig. 5[8]. 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, decoder and render buffers.

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 locally computed by the Sync Clients. Accordingly, the Sync Clients must choose the degree of the playout adjustments to achieve IDMS. On one hand, high values of the playout factor (near the maximum limit, i.e. $|\varphi_{max}|\approx 0.25$) will result in a rapid sync status. On the other hand, low values of the playout factor will originate smoother adjustments (helping to avoid noticeability), but the overall sync 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 Eqs. in (8):

$$N^{AMP}_{advanced} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu^k_i\cdot(1-\varphi_{\max})} - \frac{1}{\mu^k_i}} \right\rceil ; \quad N^{AMP}_{lagged} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu^k_i} - \frac{1}{\mu^k_i\cdot(1+\varphi_{\max})}} \right\rceil \tag{8}$$

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.

[8] 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.

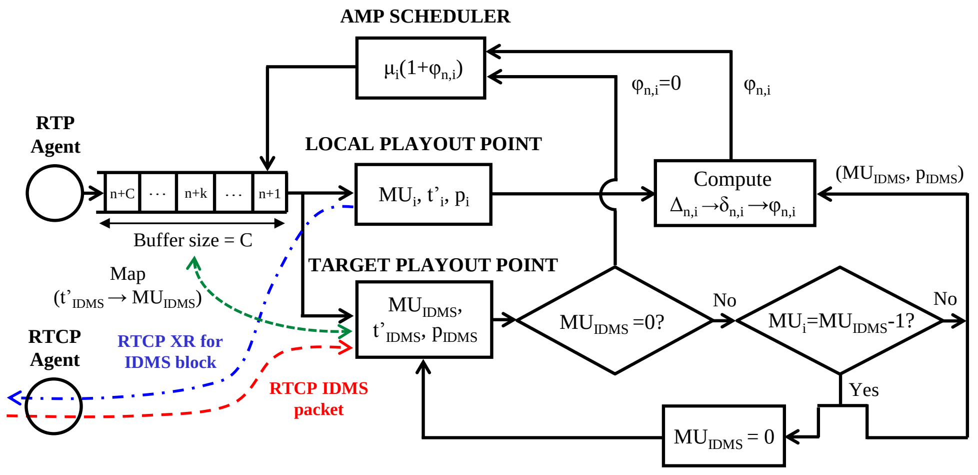

Fig. 5. Operation of the AMP Technique for IDMS.

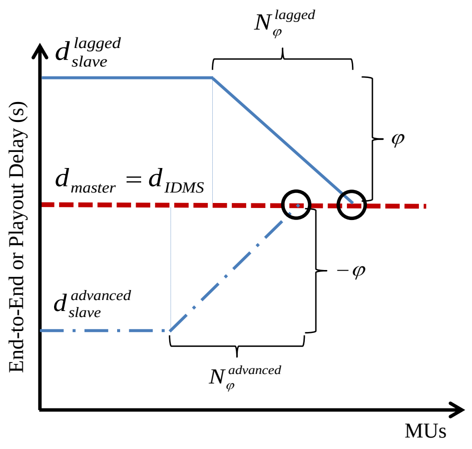

Fig. 6. Smoothed and Linear Adjustments to Acquire IDMS (DCS and M/S Scheme).

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, 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 with the IDMS target playout point (see Fig. 4), because the required adjustments are indicated to them in RTCP IDMS Settings packets.

This way, using our 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.

3.7. *Coherence Adjustment Technique (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 [3] 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. This is not a serious constraint because, anyway, the overall asynchrony in that group will 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 may not be synchronized simultaneously, and hence there will remain a residual asynchrony among them, lowering the overall sync accuracy.

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 [13] 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 sync, 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).

4. Evaluation

4.1. *Simulation Scenario and Setup*

The development, modeling and simulations were conducted using NS-2 [33]. We have tested our IDMS solution in the small scale scenario shown in Fig. 7. The simulation setup is identical to the one in [10]. This is because we want to keep continuity, as well as to show the evolution, of results with respect our work in [10], in which the preliminary version of our SMS-based IDMS solution was evaluated. As a first objective, we aim to test the correct exchange of the newly designed RTCP messages, for each one of the deployed control schemes, as well as the satisfactory performance of the newly designed algorithms and techniques for IDMS. As a second objective, we want to compare the performance of the three deployed control schemes (SMS, DCS, and M/S Scheme) in terms of several factors, such as interactivity, coherence, traffic overhead and computational load.

The simulation scenario has 7 distributed Sync Clients belonging to 2 different logical sync groups (Group 1 -G1- and Group 2 -G2-). The Media Server transmitted a media stream (in a multicast way) 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 significant e2e 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. 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.

Third, different playout rate imperfections 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* [34], in order to force higher asynchronies between the involved Sync Clients, and to test if these asynchronies were successfully handled by our IDMS solution[9]. Moreover, in order to corroborate the M/S switching capabilities of our IDMS solution (when using SMS and DCS) [5], 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.

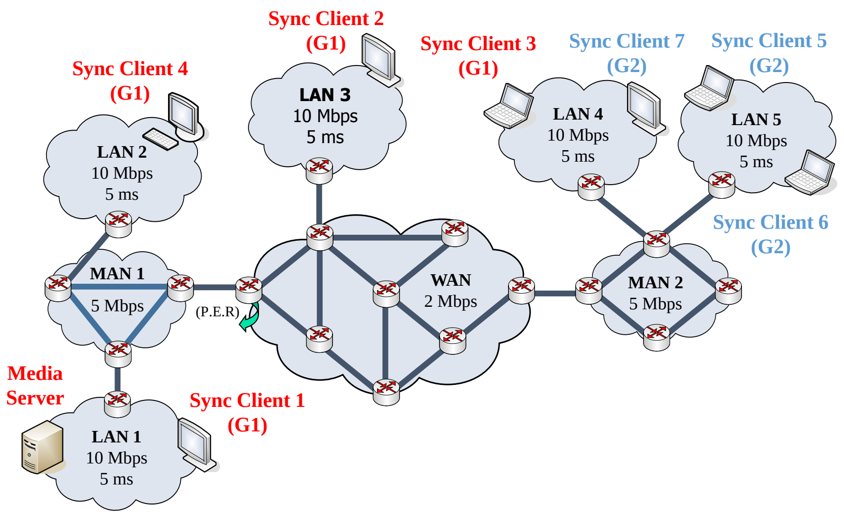

Fig. 7. Simulated Scenario.

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

[9] The effect of the delay variability and the playout rate imperfections over the media sync (especially IDMS) are described in [10].

Fourth, we included an optional functionality to simulate congestion situations at the Sync Client side (e.g., due to CPU overload, processing delays, etc.), which can cause abrupt discontinuities in the media playback and, therefore, a critical loss of sync. 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 s,* in order to force higher playout times discrepancies between the Sync Clients in that group.

With respect to wall-clock sync, 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 [3].

4.2. *Simulation Results*

4.2.1. *M/S Scheme*

The goal of this sub-section is to show the satisfactory responsiveness of M/S Scheme for IDMS in our RTCP-based solution. When using M/S Scheme, the value of τ_{max} was set to 50 ms in all the slave Sync Clients with the aim of keeping the asynchrony bounds below 100 ms. This is because, as discussed in Section 3, 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.

The playout (or e2e) 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 appreciated 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) detected 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[10] to get in sync (see zoom view). Therefore, there was a residual

[10] 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.

asynchrony of around 10 ms (i.e., τ_{max}-$1/\theta \approx$ 50-40 ms) that was not corrected. Conversely, advanced Sync Clients (SC1) had to pause their playout process every time τ_{max} was crossed in order to synchronize with the master (SC2). In such a case, advanced Sync Clients in G1 did acquire a more fine-grained sync 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 using M/S Scheme the playout adjustments are not simultaneously performed by all the slave Sync Clients. This is because each one of the Sync Clients can only compute the asynchrony between its local playout point and that of the master. That confirms the lower performance in terms of *coherence* compared to SMS [3], as shown later.

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 (see zoom view in the lower graph of 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, confirming our assumptions in Section 3.3. 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.

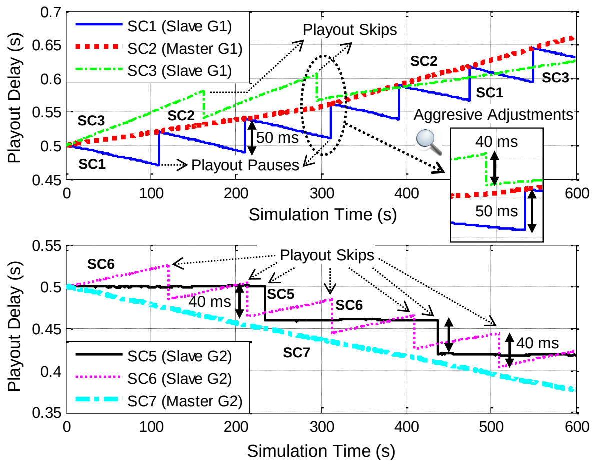

Fig. 8. Playout Delay Evolution: M/S Scheme, Aggressive Adjustments (skips & pauses).

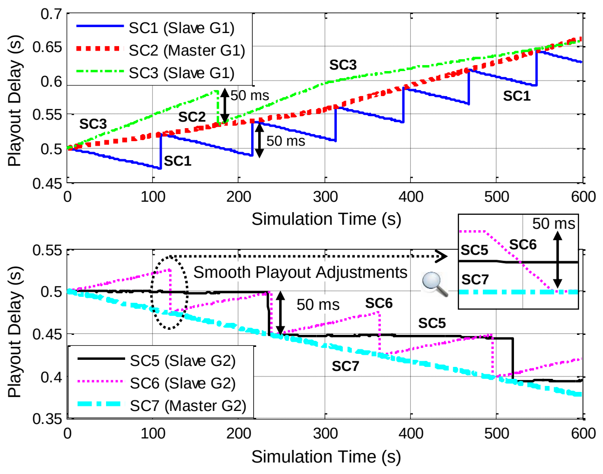

Fig. 9. Playout Delay Evolution: M/S Scheme, Smooth Adjustments (AMP).

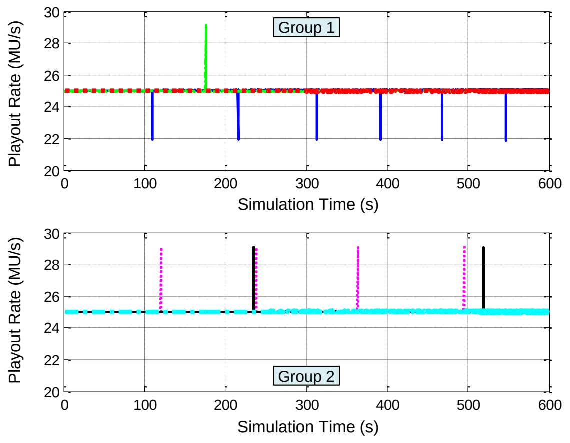

Fig. 10. Playout Rate Variation: M/S Scheme, Smooth Adjustments (AMP).

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, always keeping the playout factor within allowable bounds (i.e., $|\varphi^{k}_{max}| \leq 0.25$).

4.2.2. *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 ([5]) 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.

Despite that this policy (sync to the slowest Sync Client) can guarantee fairness, 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 sync to the fastest Sync Client policy, and using AMP as reactive technique. Here, lagged Sync Clients must speed up their playout timing every time IDMS setting instructions are received from the Sync Manager, as a consequence of the detection of an asynchrony situation.

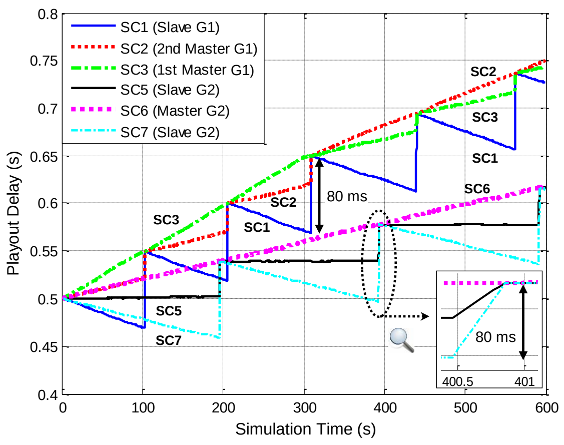

Fig. 11. Playout Delay Evolution: SMS to the Slowest with Smooth Adjustments (AMP).

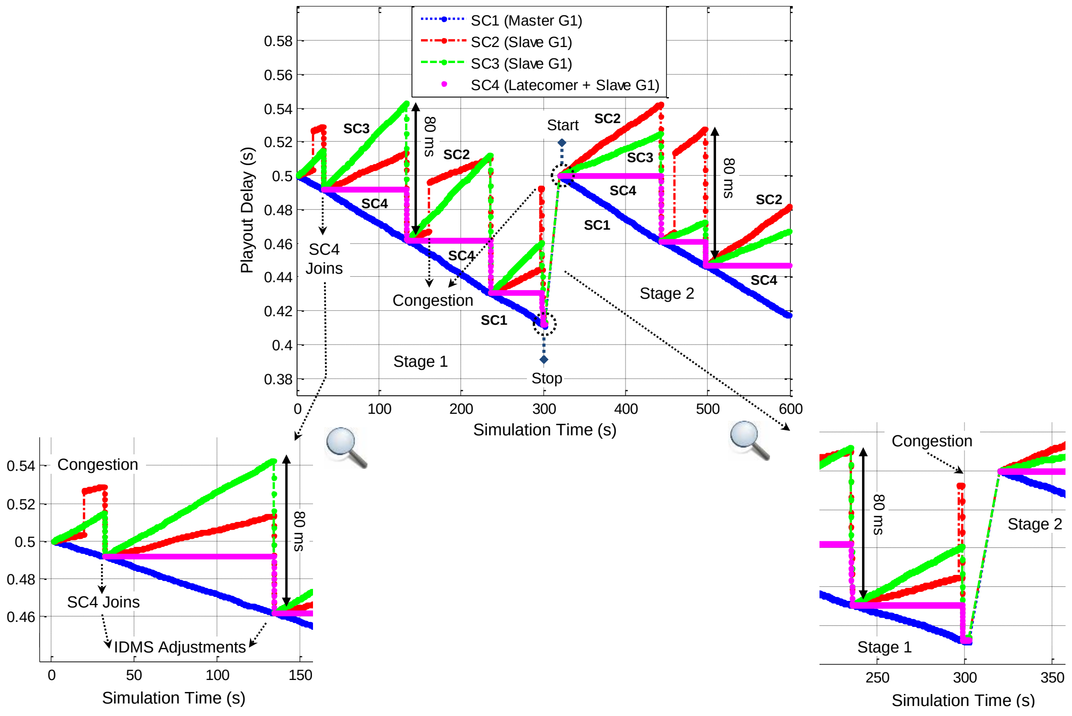

Fig. 12. SMS to the Fastest Sync Client with Coarse Sync using AMP (including Accommodation of Latecomers and Congestion Situations).

In such a case, three additional situations were considered. First, a new Sync Client (SC4) joined the on-going session at *30-th* second. Therefore, the Sync Manager sent a new RTCP IDMS Settings packet to allow that *latecomer* to acquire IDMS as soon as possible, despite that the asynchrony in that group at that moment was lower than the allowed threshold (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 appreciated in the zoom views). Third, the media session was stopped at *300-th* and re-started again at *320-th* second. It can be appreciated in the zoom views that all the Sync Clients were perfectly synchronized at the beginning of the two stages in which the session was divided (*Coarse Sync* [1, 10]), 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.

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 (speed up) their playout rate to synchronize, avoiding long-term playout discontinuities. As expected, 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 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 can minimize the occurrence of buffer underflow/overflow situations, but it is only applicable when using SMS if the Media Server and the Sync Manager are co-located. The playout delay evolution for the Sync Clients in both groups when using this policy is illustrated in Fig. 14. It can be appreciated that the Sync Clients in each group were adjusted to the ideal playout timing indicated by the Sync Manager every time an out-of-sync situation in that group was detected, resulting in a quite uniform playout delay evolution for all of them. 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 achieving IDMS. However, this policy does not take into account potential fluctuations of the e2e 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.

4.2.3. *DCS*

In this sub-section, the effectiveness of our RTCP-based IDMS solution when 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 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.

The same effect for DCS can be appreciated in Fig. 16, using the master selection policy to the mean playout point, and using AMP as reactive technique. It can be seen that the 3 Sync Clients did not always simultaneously enforce IDMS adjustments also due to the previous described situation.

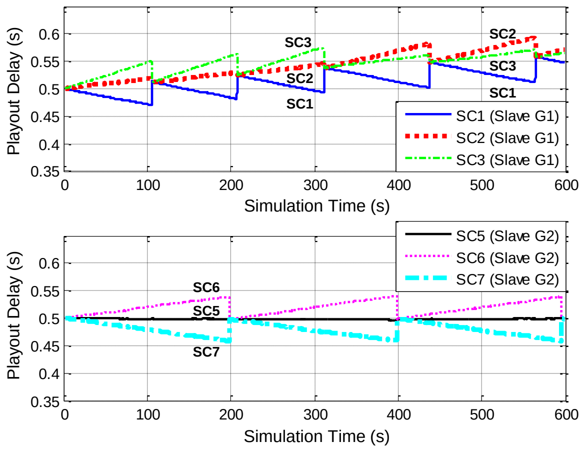

Fig. 13. Playout Delay Evolution in both groups: SMS to the Mean with AMP.

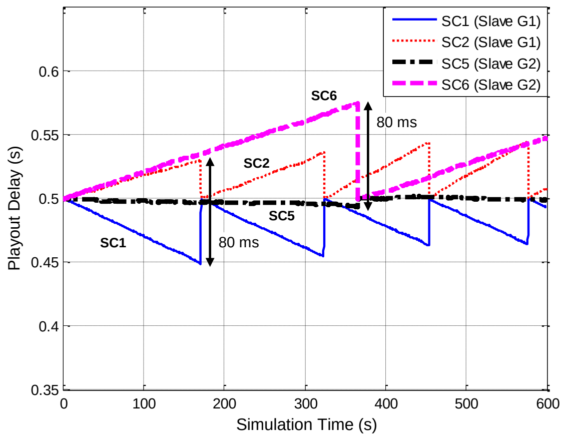

Fig. 14. Playout Delay Evolution: SMS to the Sender using AMP.

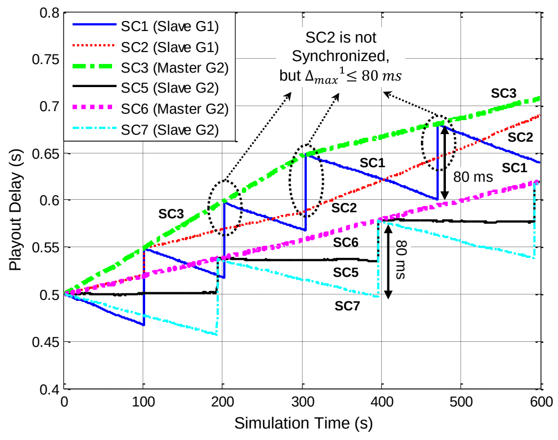

Fig. 15. Playout Delay Evolution: DCS to the Slowest, using AMP.

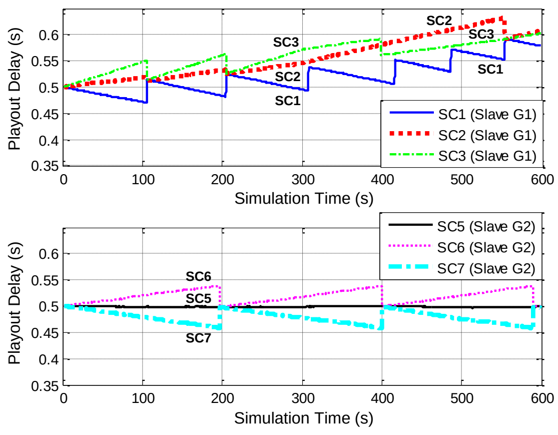

Fig. 16. Playout Delay Evolution: DCS to the Mean Playout Point, using AMP.

In order to overcome the above issue, the *coherence* adjustment technique 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 to the same IDMS reference 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. This resulted in a more fine-grained sync and a clearly better performance in terms of coherence [3].

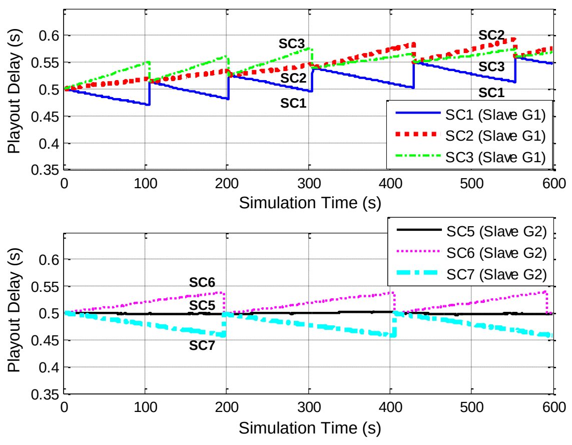

Fig. 17. Playout Delay Evolution: DCS to the Mean, using AMP + Coherence.

4.2.4. *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 because a different asynchrony threshold was set when using this scheme, as discussed earlier.

Interactivity

Figures 13 (SMS) and 17 (DCS) are almost indistinguishable. To clarify the differences among them, zoom views of the adjustments processes in both figures are shown in Figures 18 (for G1) and 19 (for G2) for both SMS (left graphs) and DCS (right graphs). It can be appreciated 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 between the Sync Clients and the Sync Manager (co-located with the Media Server). Second, the Sync Manager may not be able to send an immediate RTCP IDMS Settings packet after detecting an out-of-sync situation, since it has to adhere to the bounded RTCP timing rules, as specified in [11]. 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.

As discussed, the RTCP report interval [11], T_{RTCP}, plays a key role regarding the interactivity of our IDMS solution when using each one of the deployed control schemes. The T_{RTCP} is dynamically adjusted in each of the involved sync entities according to a local estimation of the population of the session and the available bandwidth, in order to prevent that the total amount of control traffic added by RTCP does not exceed 5 % of the allocated RTP session bandwidth [11].

In our simulated scenario, according to these RTCP timing rules [11], a delay of up to 0.6 s could be accumulated between the instant at which an out-of-sync 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. Such differences in terms of interactivity between SMS and DCS are relevant, especially for the IDMS use cases with stringent sync requirements [3].

Even though the maximum playout asynchrony in each group may slightly increase during this additional Sync Manager delay, the important fact is that the out-of-sync situation will not be repaired during this time interval. For instance, as a result of 10 simulation runs, the maximum playout asynchrony in G2 during the session using the SMS was 82.4 ms (mean value 39.4 ms) whilst the one using the DCS was 81.4 ms (mean value 38.8 ms).

It is important to emphasize that the magnitude of the Sync Manager delay would be significantly larger (up to 5 s or slightly superior) if either the minimum value for T_{RTCP} from [11] or *trr-int* attribute from [35] would have been considered in our simulation setup. The first one has not been considered since its use is dropped in [35]. Instead, *trr-int* has been introduced in [35] to restrain from sending too frequent regular RTCP packets. However, as it is an optional attribute, it may be set to zero (default value) if a specific application would benefit from a higher frequency of Regular RTCP packets, as for IDMS use cases. Furthermore, if any of these parameters had been adopted, larger delays would be introduced when gathering the IDMS reports from the Sync Clients in each one of the deployed schemes, thus also affecting to the interactivity of our IDMS solutions. This is because T_{RTCP} in each Sync Client would be larger in such a case and, therefore, asynchrony situations would be detected later.

Finally, as can be appreciated in Fig. 19, a significant advantage of using DCS for IDMS is that the 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).

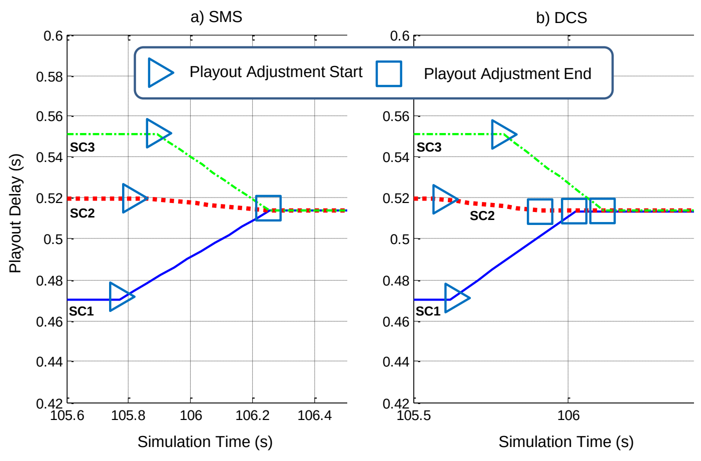

Fig. 18. Zoom View of the Playout Adjustments (AMP) in Group 1.

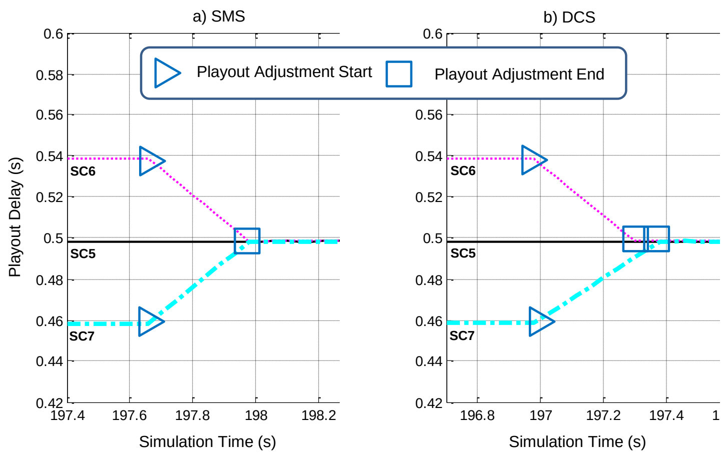

Fig. 19. Zoom View of the Playout Adjustments (AMP) in Group 2.

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, Fig. 20 shows that the playout rate was varied within tolerable limits (AMP) [30, 31] during the session lifetime in both SMS (Fig.20a) and DCS (Fig.20b), when using the master selection policy to the mean playout point. Moreover, the zoom views show that, using SMS, the playout adjustments were performed almost simultaneously in all Sync Clients (Fig. 20a), whilst this did not occur when using DCS (Fig. 20b). 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, because a single user may readily notice small delay differences (and subsequent playout adjustments), which may spoil the QoE. Therefore, the sync requirements become stricter and coherence is more relevant in such cases.

Playout Asynchrony Distribution

To complement the previous results, Fig. 21 shows the distribution of the playout asynchrony (average results of 10 simulation runs) 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 both aggressive and smooth playout adjustments.

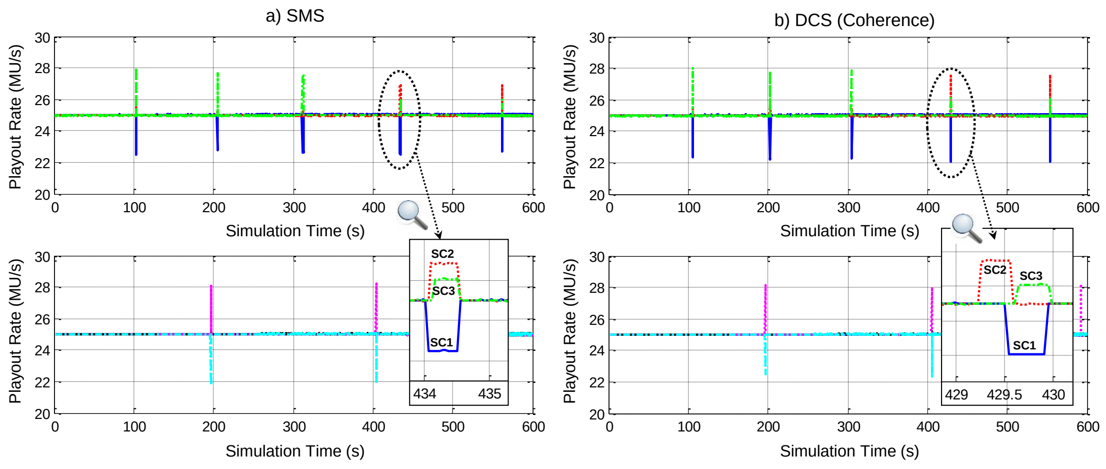

Fig. 20. Playout Rate Variation: Sync to the Mean Playout Point, using AMP.

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 [10]. 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 almost simultaneously synchronized, because of the reception of RTCP IDMS Settings packets (high performance in terms of coherence), as shown in Figures 18, 19 and 20. It can also be appreciated that this 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. As previously discussed, this percentage would have been significantly larger if any of either T^{min}_{RTCP} or *trr-int* had been adopted.

Even though the percentages of the playout asynchrony distribution depend on several factors, such as the network topology under test, available bandwidth, number of active Sync Clients and their characteristics, the values from Fig. 21 are representative for illustrative purposes, corroborating the differences between the use of the different control schemes and adjustment techniques for IDMS.

Traffic Overhead

As discussed, in standard compliant RTP/RTCP streaming scenarios [11], the total amount of RTCP traffic must be dynamically adjusted to not exceed 5 % of the allocated RTP session bandwidth. Our newly defined RTCP messages for IDMS only constitute a small subset of all existing RTCP packets and reports, and are sent in compound RTCP packets in each regular RTCP report interval [11].

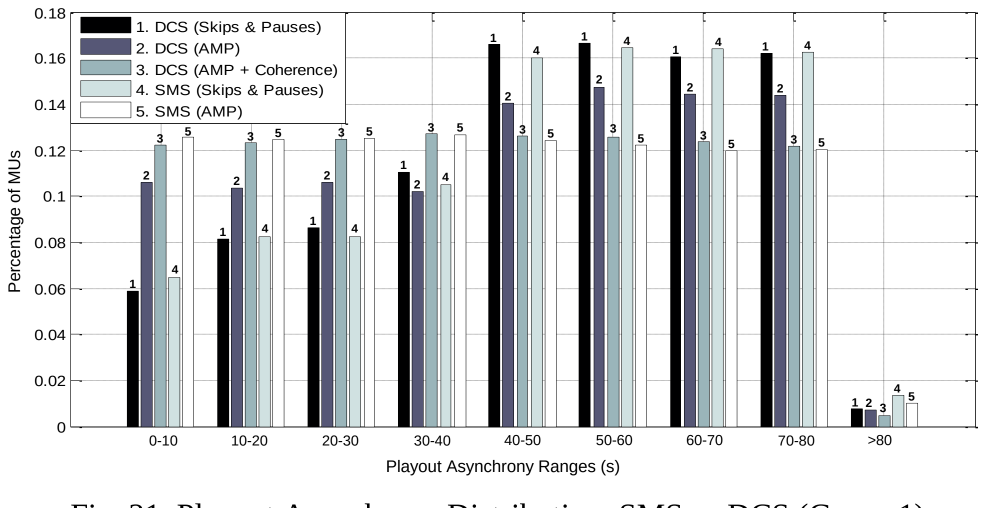

Fig. 21. Playout Asynchrony Distribution: SMS vs DCS (Group 1).

Accordingly, 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, independently on the number of active Sync Clients in the session (and on the number of involved groups), because the RTCP report interval will be dynamically adjusted to adhere to the RTCP traffic bounds [11].

In our simulated scenario, 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.). 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 in each group. 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 sync 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).

The newly defined RTCP messages in this evolved version of our IDMS solution are a bit larger than the proposed RTCP extensions in the previous version [1, 10], but this increase of size only minimally affects the frequency of RTCP reporting, according to the timing rules in [11].

Therefore, the traffic overhead for IDMS is very low, because we have not defined a new proprietary protocol, but we have taken advantage of the RTCP extension capabilities for designing our own adaptive and standardized IDMS solution.

Computational Load

Regarding the computational load, 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.

5. Conclusions and Future Work

In this work, we have presented the design and development of an adaptive, accurate and standard compliant RTP/RTCP-based IDMS solution. Moreover, because of the suitability of different control schemes for specific use cases, our IDMS solution has been extended and adapted 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. Simulation tests have shown the consistent behavior of the control schemes and the satisfactory responsiveness of our RTCP-based IDMS solution in a small scale setup. In particular, simulation results confirm prove the ability of our IDMS solutions to achieve an overall sync 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. Likewise, the results confirm the differences between IDMS control schemes, mainly in terms of coherence and interactivity.

Despite that simulation tests cannot definitively validate the performance of our 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 [3].

We believe in the relevancy of IDMS in current and future media delivery deployments. Accordingly, we have in mind several lines for future work. First, we will assess the impact of the application of the proposed IDMS techniques by implementing them in a real media framework (e.g., GStreamer [36]). This will enable us to perform real-world assessments in present-day network environments, analyzing the effects on the QoE of different levels of out-of-sync situations and how they are handled/avoided by using our IDMS solution. Second, we plan to design further RTCP extensions to enable the triggering of dynamic IDMS adjustments based on the instantaneous importance of the media content being delivered, as done in other works for inter-stream sync purposes (e.g. [2]). Third, we plan to further investigate on AMP for IDMS. On one hand, we will examine the effectiveness of our AMP technique when using modern H.264 SVC (Scalable Video Coding) codecs with increased rate variability [37], and the implications on various aspects, such as delay, buffer settings, computational load or sync accuracy. On the other hand, we plan to design advanced adjustment techniques (e.g., non-linear, content-aware) with the objective of minimizing abrupt changes in the media playout rate. Finally, we need to extend our studies with a thorough analysis of the behavior of our IDMS solution in large-scale scenarios, with lots of Sync Clients joining and leaving the session. This will determine the relevancy, stability and scalability of our IDMS solution, and if further mechanisms need to be designed and adopted to enhance its performance in such situations.

ACKNOWLEDGMENT

This work has been financed, partially, by Universitat Politècnica de València (UPV), under its R&D Support Program in PAID-01-10. 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] 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), pp. 285-319, June 2008.

[2] I. H. Elhajj, N. B. Dargham, N. Xi, and Y. Jia, "Real-time adaptive content-based synchronization of multimedia streams", Advances on MultiMedia Journal, Article 4 (January 2011).

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

[4] 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 (7), pp. 370-377, August 2011.

[5] F. Boronat, J. Lloret, and M. García, "Multimedia group and inter-stream synchronization techniques: A comparative study", Information Systems, Vol. (34), 1, pp. 108-131, March 2009.

[6] Z. Huang, K. Nahrstedt, and R. Steinmetz, "Evolution of temporal multimedia synchronization principles: A historical viewpoint", ACM TOMCCAP, Vol. 9, 1s, Article 34, October 2013.

[7] O. van Deventer, H. Stokking, O. Niamut, F. Walraven, V. Klos, Advanced Interactive Television Service Require Synchronization, IWSSIP 2008, Bratislava, June 2008.

[8] D. A. Shamma, M. Bastea-Forte, N. Joubert, Y. Liu, "Enhancing online personal connections through synchronized sharing of online video, ACM CHI'08 Extended Abstracts, Florence (2008).

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

[10] 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.

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

[12] 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.

[13] R. van Brandenburg, H. Stokking, O. van Deventer, F. Boronat, M. Montagud, K. Gross, "RTCP for inter-destination media synchronization", draft-ietf-avtcore-idms-13, IETF Audio/Video Transport Core Maintenance Working Group, Internet Draft, August 2013.

[14] 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.

[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] 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

[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 (1), February 2004.

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

[23] 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.

[24] 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 Real time Interactive Systems, SEARIS 2010, Massachussets, USA, March 2010.

[25] M. Montagud, F. Boronat, "RTP/RTCP and Media Sync: A Review and Discussion of Future Work", MediaSync Workshop, Nantes (France), October 2013

[26] WebRTC 1.0: Real-time Communication Between Browsers, World Wide Web Consortium (W3C) Working Group, http://dev.w3.org/2011/webrtc/editor/webrtc.html, Last Accessed 2013-08-02.

[27] Real-Time Communication in WEB-browsers (RTCWeb), Active Working Group within the IETF, http://tools.ietf.org/wg/rtcweb/, Last Accessed in 2013-08-02.

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

[29] A. Williams, K. Gross, R. van Brandenburg, H. Stokking, "RTP Clock Source Signalling", draft-williams-avtcore-clksrc-11, IETF AVTCORE Group, Internet Draft, March 2014.

[30] 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.

[31] Y. Su, Y. Yang, M. Lu, H. Chen, "Smooth Control of Adaptive Media Playout for Video Streaming", *IEEE Transactions on Multimedia*, Vol. 1 (7), pp. 1331-1339, November 2009.

[32] Y. Ishibashi, S. Tasaka, H. Ogawa, "Media Synchronization Quality of Reactive Control Schemes", *IEICE Transactions on Communications*, Vol.E86-B (10), pp. 3103-3113, October 2003.

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

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

[35] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

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

[37] G. Van der Auwera, and M. Reisslen, "Implications of Smoothing on Statistical Multiplexing of H.264/AVC and SVC Video Streams", IEEE Trans. On Broadcasting, Vol. 55 (3), September 2009.