Early Event-Driven (EED) RTCP Feedback for Rapid IDMS

Mario Montagud, Fernando Boronat
Universitat Politècnica de València (UPV)
Calle Paranimf, 1, 46730
Grao de Gandia (Valencia), SPAIN
+34 962849341
{mamontor@posgrado, fboronat@dcom}.upv.es

Hans Stokking
TNO
Netherlands Organisation for Applied Scientific Research
Brassersplein 2, Delft (the Netherlands)
+31 88 86 67278
hans.stokking@tno.nl

ABSTRACT
Inter-Destination Media Synchronization (IDMS) is essential in the emerging media consumption paradigm, which is radically evolving from passive and isolated services towards dynamic and interactive group shared experiences. This paper concentrates on improving a standardized RTP/RTCP-based solution for IDMS. In particular, novel Early Event-Driven (EED) RTCP feedback reporting mechanisms are designed to overcome latency issues and to enable higher flexibility, dynamism and accuracy when using RTP/RTCP for IDMS. The faster reaction on dynamic situations (e.g., detection of asynchrony or channel change delays) and a finer granularity for synchronizing media-related events, while preserving the RTCP bandwidth bounds, are validated through simulation tests.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems.

General Terms
Performance, Design, Experimentation.

Keywords
Event-Driven, IDMS, RTP/RTCP, Simulation, Synchronization.

1. INTRODUCTION
Nowadays, we are witnessing a real transition from physical togetherness towards networked togetherness around media content. Traditionally, users have gathered at a single location for consuming media (e.g., for watching TV content). The typical scenario is a group of friends watching a live football match at a friend's home. But recent advances on media streaming and on social networking, in conjunction with the proliferation of connected devices, lead to a new media consumption landscape. Novel forms of shared media experiences are gaining momentum [1], allowing geographically distributed users to socially interact (e.g., using text, audio or video chat, or combinations thereof) within the context of simultaneous content consumption. The co-located friends in the above example can now watch the football match from their own home, while being able to converse, discuss about its evolution, and cheer together when goals are scored.

MM'13, October 21–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2502081.2502114

However, realizing those shared interactive services faces a lot of challenges [2]. In particular, this paper focuses on the provisioning of synchronized playout in all the involved consumer devices. This process is commonly referred to as Inter-Destination Media Synchronization (IDMS), and its relevancy is increasing in a large number of use cases, such as networked multi-player games, synchronous e-learning, or Social TV [1]. For instance, in the above *"watching apart together"* scenario, being aware of a goal through the cheering of a friend via the chat channel, before the goal sequence is displayed on the local screen, can be very frustrating and would spoil the shared experience [3].

The main technological barrier for IDMS is the end to end (e2e) delay variability when delivering media to distributed clients. Different sources of delay in the distribution chain can contribute to that variability, which can range from few milliseconds up to several seconds, depending on the technology in use [1, 3-5]. For instance, the measurements in [5] showed that the e2e delays for the shortest and longest paths of an IPTV scenario can differ up to 6 s. Without intervention, such different delays lead to differences in playout timings, thus preventing seamless and coherent interactions in these networked environments.

The need for IDMS for supporting interactive shared TV experiences has been reported in previous works [3, 6, 7]. For instance, controlled experimental setups have analyzed the effect of de-synchronization on the Quality of Experience (QoE) in Social TV scenarios [7, 3]. In [7], distributed users watched a quiz show, while interacting via voice and text chat. It was concluded that delay differences up to 1 s might not be noticeable by users, but differences over 2 s really become annoying for most of them (i.e., both text and voice chatters). Similar results were obtained in [3] by recreating the football watching experience. These user perception tests provide initial empirical evidence that actual delay differences lead to a severe QoE degradation. Moreover, Social TV is not, by far, the most restrictive IDMS use case, and other scenarios require stringent synchronization (sync, hereafter) levels [1]. Consequently, this motivates the development of adaptive and accurate IDMS solutions to compensate such e2e delay variability.

Due to the increased relevancy of IDMS, several works have addressed this topic up to now. Some illustrative papers discuss a number of use cases [1], report on working prototype implementations (e.g., [2, 8-11]), and provide a taxonomy of existing IDMS solutions [12]. Among them, a noteworthy approach consists of extending the RTCP capabilities [13] to provide useful feedback information for IDMS (e.g., [10, 11]). Using RTP/RTCP [13] for IDMS is advantageous due to several reasons. First, the use of RTCP as a feedback channel allows for continuous monitoring and control processes of the overall IDMS timings. This differs from other basic solutions that uniquely rely on synchronizing specific control events. Accordingly, these

solutions provide a coarse sync process, with lower accuracy, because of the continuous and unpredictable e2e delay variability during a media session's lifetime. Second, the IDMS problem can be tackled above the transport layer, based on the current presentation times of each client. This way, the e2e delay variability can be compensated for. This allows the achievement of more accurate sync levels than the ones in specific solutions based only on compensating the network delay variability, because the media content is also significantly delayed in different manners at the client side [1, 3-5]. Third, the current standardization efforts within the ETSI [14] and IETF [15] to provide RTP/RTCP-based technology for IDMS will help to ensure interoperability and to promote deployment in real environments (e.g., in IPTV). In contrast, most of the existing solutions (surveyed in [12]) define proprietary protocols that may increase the network load and make compatibility between third-party implementations more difficult. Further benefits of using RTP/RTCP for IDMS include: i) the widespread use and support of those protocols; ii) the inclusion of timestamps in the media delivery units (i.e., RTP packets); iii) the support for intra- and inter-stream sync; iv) the adaptability and scalability of RTCP (see next Section); v) the ability of negotiating the use of common wall-clock sources [16]; vi) the inherent rate adaptive mechanisms; etc.

Acknowledging the suitability of and last advancements on RTP/RTCP-based technology for IDMS, we still foresee a key limitation: the proposed RTCP messages for IDMS [15] are exchanged in a pre-scheduled manner, uniquely based on preserving the allowed traffic bounds specified in [13]. This cannot provide efficient IDMS control, because there is no provisioning for timely feedback that would allow to repair or to manage dynamic events of interest close to their occurrence.

The goal of this paper is to overcome those issues, by making the use of the RTCP channel for IDMS more efficient. Novel standard compliant RTCP reporting rules and messages, which in conjunction we call Early Event-Driven (EED) RTCP Feedback, are presented for achieving a more rapid, flexible, dynamic and accurate IDMS control, while being backward compatible with standardized RTP/RTCP solutions [13-18], and still adhering to the RTCP traffic bounds specified in [13]. In particular, the EED RTCP Feedback for IDMS provides the three following advantages. First, asynchrony situations can be repaired earlier than using the Regular RTCP reporting rules [13] adopted up to now. Second, the use of EED RTCP Feedback enables the concurrent presentation of dynamic media-related events in a fine-grained synchronized way with the piece of content they refer to. Third, the latency from the instant at which receivers join RTP (multicast) sessions until they achieve IDMS (referred to as IDMS latency) can be significantly reduced. The proposed RTCP extensions for IDMS are applicable to and can have a potentially high impact on a wide spectrum of scenarios with demanding IDMS characteristics [1], such as IPTV, networked multi-player games, synchronous e-learning, etc.

Through simulation tests we aim to provide evidence of the ability of the proposed EED RTCP Feedback to achieve faster reaction and more accurate responsiveness to dynamic situations in IDMS-enabled sessions, thus overcoming important limitations of existing RTP/RTCP-based IDMS solutions [10, 11, 15].

The remainder of the paper is structured as follows. The next section presents an overview of the standardized RTCP timing rules. Section 3 discusses related work. Then, the novel aspects of the EED RTCP Feedback for enhancing the IDMS performance are presented in Section 4. Section 5 gives performance results, and, finally, in Section 6 we draw some conclusions, summarize our contributions and discuss future work.

2. STANDARD RTCP REPORTING RULES

In this section, an overview of the RTCP reporting rules specified in different IETF standards is provided. This is important to help understanding (the benefits of) the EED RTCP Feedback for IDMS proposed in this paper.

2.1 Regular RTCP Feedback (RFC 3550)

The participants of an RTP Session regularly exchange RTCP reports to inform mainly about Quality of Service (QoS) statistics [13]. On the one hand, a low frequency of feedback reporting can lead to faulty behavior owing to outdated statistics. On the other hand, excessive reports can be redundant and cause unnecessary control traffic. Also, if the RTCP reports were exchanged at a constant rate, the control traffic would grow linearly with the number of participants. Therefore, a trade-off between up-to-date information and the amount of control traffic must be met. To do so, the RTCP feedback rate must be dynamically adjusted according to the estimated population of the session.

The total amount of control traffic added by RTCP should be limited to a small (so that the primary function of media data transport is not impaired) and known (so that each participant can independently calculate its share) fraction of the allocated RTP session bandwidth ($BW_{session}$). A fraction of 5 % is recommended in [13]. 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 and the remaining can be consumed by receivers. Otherwise, the RTCP bandwidth is equally shared between senders and receivers. Accordingly, the RTCP report interval is deterministically computed, $T^{3550}_{RTCP\_d}$, based on the allocated $BW_{session}$, the average size of all received and sent RTCP packets ($RTCP_{size}$), the number of participants in the session and their role (senders or receivers). These formulas are in Eqs. (a) of Fig. 1[1].

However, $T^{3550}_{RTCP\_d}$ should have a lower bound to avoid having bursts of RTCP packets. The recommended value in [13] for the minimum interval, $T^{3550}_{RTCP\_d\_min}$, is 5 s (see Eq. (b1) of Fig. 1). In some cases (e.g., if the data rate is high and the application demands more frequent RTCP reports), an implementation may scale $T^{3550}_{RTCP\_d\_min}$ to a smaller value given by 360 divided by $BW_{session}$ (in kbps), as shown in Eq. (b2) of Fig. 1. This yields an interval smaller than 5 s when $BW_{session}$ becomes greater than 72 *kbps*. Accordingly, the minimum value between $T^{3550}_{RTCP\_d}$ and the selected option for $T^{3550}_{RTCP\_d\_min}$, will be used for the RTCP report interval, as shown in Eq. (c) of Fig. 1. After that, the interval between RTCP packets is varied randomly over the range [0.5, 1.5] times that minimum RTCP report interval, as shown in Eq. (d) of Fig. 1, to prevent floods of RTCP reports.

[1] We have placed all the Eqs. in Fig.1 to provide a schematic view of the stepwise calculation process for the RTCP report interval.

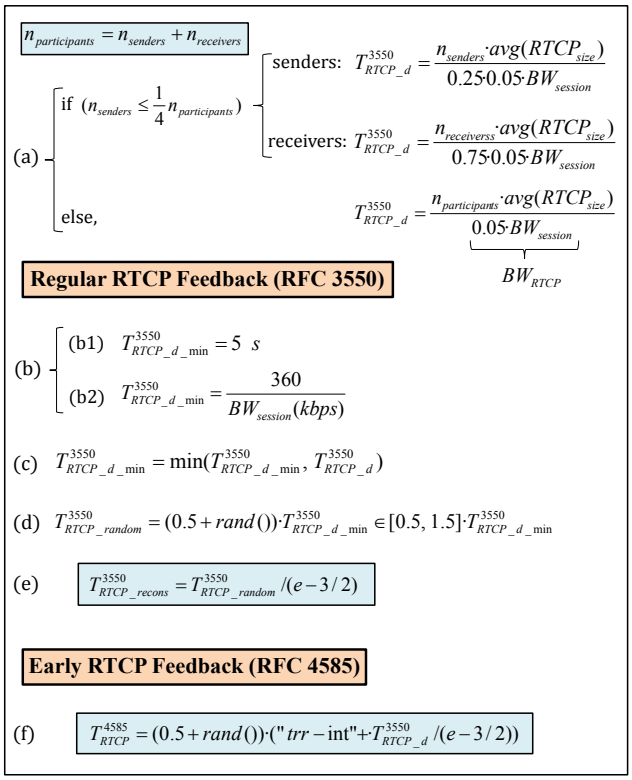


Figure 1. Calculation Steps of the RTCP Report Interval.

Additionally, *"timer reconsideration"* algorithms are introduced in [13] to allow for a more rapid adaptation of the RTCP report interval in large-scale sessions, where the membership can largely vary. To compensate for the fact that the *"timer reconsideration*" algorithms converge to a lower value than the intended average RTCP bandwidth, the computed report interval is finally divided by e-3/2=1.21828, as shown in Eq. (e) of Fig. 1.

2.2 Early RTCP Feedback (RFC 4585)

In [17], further RTCP reporting mechanisms are specified to enable receivers to provide, statistically, more immediate RTCP feedback to the senders. This Early RTCP Feedback profile, which is known as RTP Audio-Visual Profile with Feedback (RTP/AVPF), allows for short-term adaptation and efficient feedback-based repairing mechanisms to be implemented, while maintaining the RTCP bandwidth constraints and preserving scalability to large groups. The RTCP report interval specified in [13] is denoted as Regular RTCP interval in [17]. In addition, it is specified in [17] that RTCP reports can be reported earlier than the next scheduled Regular RTCP transmission time if a receiver detects the need to inform about events of interest about the media stream (e.g., picture or slice loss) close to their occurrence[2].

The reporting rules for Regular RTCP packets in [17] are similar than the ones in [13]. However, $T^{3550}_{RTCP\_d\_min}$ is dropped in [17]. Instead, an optional attribute, called *trr-int*, is specified as an offset parameter (in ms) to $T^{3550}_{RTCP\_d}$, as shown in Eq. (f) of Fig.1. Note that providing *trr-int* as an independent variable is meant to minimize the frequency of Regular RTCP packets (i.e., saving RTCP bandwidth), while allowing more flexibility to transmit Early RTCP packets (i.e., using the saved RTCP bandwidth) in response to dynamic events. This could not be achieved by reducing the overall RTCP bandwidth, because the Early RTCP packets would be affected as well. Values between 4 and 5 s for *trr-int* are recommended in [17] to assure interworking with RTP entities only using Regular RTCP Feedback [13]. However, as *trr-int* 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. In such a case, the only difference between Regular [13] and Early RTCP Feedback [17] for transmitting Regular RTCP packets resides in the minimum value for the report interval, which is dropped in [17].

2.3 Rapid Inter-Stream Sync (RFC 6051)

In multimedia streaming services, the inter-stream sync delay refers to the time difference between the instant at which a user joins a multicast session, probably involving different media (e.g., audio and video, or when using layered and/or multi-description codecs) carried in separate streams, and the instant at which these correlated streams can be synchronously presented to that user [18]. The aim in RFC 6051 [18] is to minimize the inter-stream sync delay when using RTP/RTCP for media delivery. The motivation is that a receiver cannot synchronize playout until a compound RTCP packet, including a Source Description or SDES packet (source identification) and a Sender Report or SR (timing correlation parameters) [13], is received for all the involved RTP sessions. If there is no packet loss, this gives an expected delay equal to the average time for receiving the first RTCP packet from the RTP Session with the longest RTCP report interval[3].

RFC 6051 [18] introduces three backward compatible extensions to RTP/RTCP [13] to reduce the inter-stream sync delay. First, the RTCP timing rules are updated to allow Single Source Multicast (SSM) senders [19] the immediate transmission of an initial RTCP packet upon joining each RTP session in a multimedia session. The rationale for not allowing the transmission of immediate RTCP packets to SSM receivers is to avoid feedback implosion if lots of receivers join the session almost simultaneously. Second, a new RTP/AVPF transport layer feedback message [17] is defined to allow receivers to request the generation of an Early RTCP SR from the media sender. This enables rapid (re-)sync in case that an RTCP SR has not been received for a long period (e.g., packet loss), or to allow latecomers to achieve inter-stream sync as soon as possible. Finally, new RTP header extensions are defined to enable the inclusion of in-band sync metadata with RTP data packets.

3. RELATED WORK

In this section, some implementations making use of the above-described standardized RTP/RTCP reporting rules to increase the video quality (Section 3.1) and to reduce the inter-stream sync delay (Section 3.2) are presented. As these proofs of concept provided better QoE in streaming services, this motivates the design and development of novel EED reporting mechanisms (Section 4) to enhance the responsiveness of existing RTP/RTCP-

[2] A suppression mechanism is adopted, in which receivers wait for a random dithering interval to avoid RTCP feedback implosion (i.e., lots of receivers reporting on the same event) [17].

[3] Note that the inter-stream sync delay depends on the specific instant at which a user joins the multimedia session or each RTP session (e.g., the user may first receive the RTCP packets from the RTP session with the longest RTCP interval), as well as on the impact of the randomization processes in all the involved RTP sessions.

based solutions for IDMS using a centralized approach (Section 3.3). To conclude this section, previous works on event-based IDMS are briefly described in Section 3.4.

3.1 Use of Early RTCP Feedback

The work in [20] showed that the proposed RTCP extensions in [17], in conjunction with temporal video adaptation mechanisms, resulted in a significant video quality gain under lossy conditions. Also, the use of RTP/AVPF [17] allowed a better performance of a proposed error-resilient video coding technique in [21].

3.2 Reduction of Zapping and Sync Delay

Reducing channel-change (i.e., zapping) delays is a major concern for IPTV services. This involves optimizing several components of IPTV systems [22], such as media (trans-)coding, multicast (join/leave) procedures, predictive tuning methods, acquisition of the necessary reference information from the stream to start its consumption, buffering techniques, etc. Up to now, several solutions have been devised to overcome the zapping latency by using RTP/RTCP mechanisms (e.g., [22-24]). However, in conjunction with the above techniques, the RTCP timing rules from [18] should be provided to enable rapid (inter-stream) sync. This is because media will not be played out until the associated sync info is available, and the sync process should not contribute to further increase the channel-change delay. As an example, the works in [23, 24] made use of a rapid acquisition technique (by employing an auxiliary retransmission server), combined with enhanced RTCP reporting mechanisms, to decrease zapping (and sync) delays when joining on-going RTP multicast sessions.

3.3 RTP/RTCP for IDMS

Previous studies have shown the feasibility of RTP/RTCP for IDMS, in both real [10] and simulated environments [11]. Based on the initial idea in [10], standardization processes have been undertaken to provide RTP/RTCP-based technology for IDMS [14, 15]. Two additional RTCP messages for IDMS have been specified in [15]. First, an RTCP XR (Extended Report) block for IDMS, called IDMS report, enables clients in the IDMS session (i.e., Sync Clients) to provide feedback about reception and/or presentation times for specific RTP packets. Second, a new RTCP IDMS packet type, called IDMS Settings packet, is used to provide guidance on when to play out the media.

The above RTP/RTCP-based IDMS solutions adopted a centralized Sync Maestro Scheme (SMS) as the communication process between the involved sync entities [1]. The operation of SMS is sketched in Fig. 2. First, each Sync Client sends (unicast) IDMS reports to a single Sync Manager. Based on the collected IDMS reports, the Sync Manager computes the delay differences among the Sync Clients and, if the detected asynchrony exceeds a specific threshold, it will send (multicast) an IDMS Settings packet to notify the Sync Clients of the required adjustments to achieve IDMS.

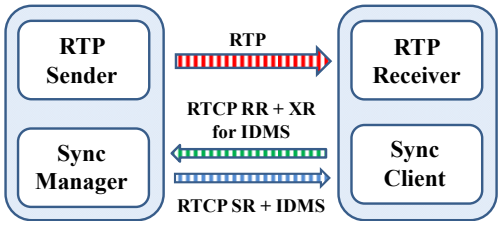


Figure 2. RTP/RTCP-based IDMS Solution Using SMS.

An exhaustive qualitative comparison between the existing control schemes for IDMS is provided in [1]. This study pointed out that SMS is, in general, best suited for IDMS. Concretely, SMS is preferable in such use cases in which consistency, coherence and security aspects must be ensured. Likewise, SMS can provide a satisfactory responsiveness in terms of flexibility, traffic overhead, causality and fairness. However, two main downsides of using SMS for IDMS were identified: scalability and interactivity. On the one hand, scalability-wise there are no significant differences between a centralized and a distributed architecture with respect to the IDMS control. This is due to the fact that in both approaches all the IDMS reports converge either to the Sync Manager (in SMS) or to each one of the involved Sync Clients (in a distributed architecture). Moreover, two mechanisms help to partially mitigate scalability issues of SMS when using RTP/RTCP for IDMS. First, the RTCP report interval is dynamically adjusted according to the number of active Sync Clients and the available bandwidth in the session (see Section 2.1) [13]. Second, the Sync Clients can be divided into logical groups, which facilitates the IDMS management to the Sync Manager [11, 15]. On the other hand, interactivity is an especially relevant limitation of SMS when using an RTP/RTCP-based IDMS solution. This is because of the required bidirectional communication process to exchange the IDMS information (see Fig. 2): first, the Sync Manager must collect the IDMS reports from all the Sync Clients; second, the Sync Manager must adhere to bounded timing rules [13] to be able to send a new RTCP IDMS Settings packet; and third, this packet has to be received by all the Sync Clients. The interactivity constraints could limit the implementation of an SMS-based IDMS solution in those use cases in which very high sync granularity and timely responsiveness to dynamic events (e.g., avoidance of asynchrony situations, rapid channel change delays, etc.) are required [1].

3.4 Event-Based Sync

Two different approaches for media sync can be distinguished: axis-based and event-based [12]. On the one hand, axis-based solutions aim to continuously align the presentation of media streams along either a virtual or a wall-clock timeline axis. On the other hand, event-based solutions handle the sync of media streams over a discrete set of reference points or events. These events can be either specific occurrences in time within the media stream or state modifications (e.g., user generated actions), either sporadic or periodic, and either inserted into the media stream or sent in parallel using another communication channel.

Event-based sync solutions have been traditionally used in networked games [12] and in Collaborative Virtual Environments [25] to keep a consistent shared session. In [2], a popular event-based sync algorithm for networked games [8] was adapted to be used for shared video watching. The goal was to achieve a concurrent sync of specific user actions at all the clients. The algorithm consists of two parts: i) *local lag* to compensate for short term inconsistencies; and ii) *time warp* to undo inconsistencies that may still occur due to extreme delay variability.

Mostly, event-based sync solutions have adopted a distributed receiver-based approach [1, 12, 25]. This requires the availability of a multicast channel among all involved users to be able to exchange events, which is not feasible when using specific streaming technologies, such as Source Specific Multicast (SSM) with unicast feedback [19], because only the Distribution Source

can transmit data in a multicast way. This is not an issue when using SMS, because the feedback reports for IDMS are sent by the Sync Clients in a unicast way to the Sync Manager.

Moreover, in this work we are not dealing with updates of the media stream content, state or position by the Sync Clients, but with dynamic events in the session (e.g., detected or triggered by the Sync Manager) that need to be handled in a timely fashion. The intention is to take advantage of the flexibility, dynamism and interactivity features provided by event-based (distributed) approaches, while still adhering to the benefits of using an axis-based (centralized) solution for IDMS [1, 12].

4. EED RTCP FEEDBACK FOR IDMS

To date, the existing RTP/RTCP-based IDMS solutions using SMS (e.g., [10, 11]) have used Regular RTCP Feedback [13]. Consequently, there may be a variable time lag (according to Eqs. in Fig. 1) between detecting an event and being able to send an appropriate RTCP packet to handle it. Furthermore, the RTCP reports may even not be received at the target side, since RTCP is sent over UDP and thus it is not a reliable control channel. This section describes the novel EED RTCP reporting mechanisms we propose to overcome these issues, thus enabling higher flexibility, dynamism, interactivity and accuracy when using SMS for IDMS.

4.1 Immediate Initial RTCP IDMS Settings

The same rationale for reducing the inter-stream sync delay in [18] (Section 2.3) can be used for IDMS purposes. When using SMS in an RTP/RTCP-based solution, it would also be desirable to transmit a nearly-immediate[4] RTCP IDMS Settings packet by the Sync Manager upon establishing a multimedia session. This would ensure a reduction of the IDMS latency experienced by the Sync Clients in a shared session.

If the Sync Manager is integrated within the Media Server (Fig. 2), it must send the IDMS Settings packet in parallel with the initial RTP data packets. If the Sync Manager is co-located within a Sync Client or a third party entity, it must send the IDMS Settings packet as soon as it receives the initial RTP data packets from the Media Server. In either case, as the Sync Manager is a single centralized RTP entity, it must also be allowed to transmit Early RTCP packets [18]. This way, the Sync Clients can start synchronously consuming the media earlier.

4.2 Dynamic EED RTCP IDMS Settings

During the media session's lifetime, if Regular RTCP Feedback is used, the Sync Manager may have to wait a nearly-complete RTCP reporting interval to be able to send a new compound RTCP packet (including an IDMS Settings packet) after detecting an out-of-sync situation, which might potentially take several seconds (up to 5 s or even more) [13]. This is illustrated in Fig. 3. In such a case, if an out-of-sync situation is detected just after the transmission of an RTCP packet (at instant $t_{r(1)}$), the next RTCP packet cannot be sent until the next randomized RTCP transmission time (at instant $t_{r(2)}$). The figure shows the worst case, in which the randomized RTCP report interval is near the upper limit, i.e.:

$$t_{r(2)} - t_{r(1)} \approx 1.5 \cdot [\, t_{d(2)} - t_{r(1)} \,] \quad (1)$$

Where:

- $t_{d(n)}$: *n-th* Scheduled (Deterministic) RTCP Transmission Time.
- $t_{r(n)}$: *n-th* Real (Randomized) RTCP Transmission Time.

Therefore, the contribution of the Sync Manager delay (i.e., the time interval since an event is detected and an IDMS Settings packet is sent) to the total IDMS latency in case of an out-of-sync situation (see Fig. 3) becomes a serious barrier for those use cases requiring stringent sync levels (e.g., networked video walls, networked loudspeakers, or networked games) [1].

To overcome this issue, the Sync Manager is allowed to send Early RTCP packets as a response to dynamic events. Figure 4 illustrates this process, in which an IDMS Settings packet is sent just after the detection of an event, despite that this moment is earlier than the next scheduled Regular RTCP transmission time (faster/immediate reaction of the Sync Manager). Consequently, the IDMS latency is significantly reduced, mainly due to the fact that the Sync Manager delay has been minimized.

Note that if *trr-int* is set to zero, only one Early RTCP packet can be transmitted between two consecutive Regular RTCP packets in order to preserve the RTCP traffic bounds [17]. It means that an Early RTCP packet can only be sent if the previous transmitted RTCP packet was a Regular RTCP packet. Hence, after sending an Early RTCP packet, the RTCP reporting engine must schedule the sending time for the next RTCP packet by delaying (i.e., skipping) one more Regular RTCP report interval (see dotted arrows in Fig. 4).

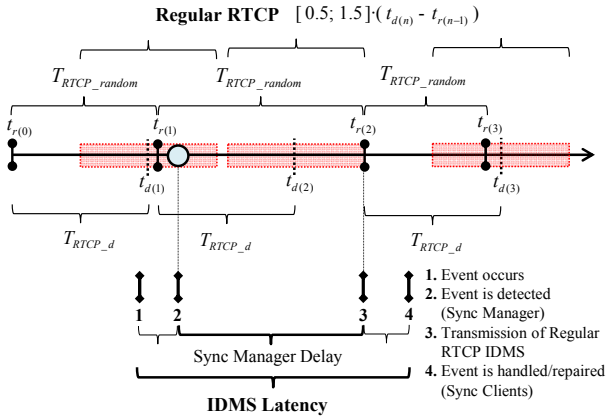


Figure 3. Regular RTCP Feedback.

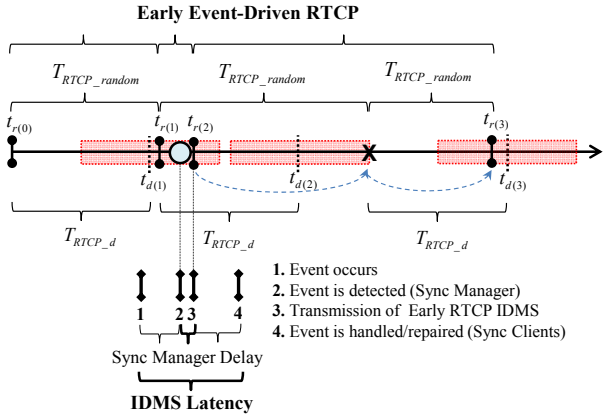


Figure 4. Early Event-Driven (EED) RTCP Feedback.

[4] Note that in this work, the terms (nearly-)immediate, close-to-instant and Early are used as synonymous. This is because the Sync Manager is a single centralized entity in the media session, and the Early RTCP packets can be sent immediately by this entity without requiring a contention algorithm, as required for receivers in [17].

This can also be very useful to provide playout hints for specific events that must be presented to all the involved users in a fine-grained synchronized way with the piece of content they refer to. Those events can be media-related events whose timing can be known in advance (e.g., commercials, start of the match in a sports event, etc.); but if the RTP/RTCP implementation somehow has direct access to the application layer, the events' timing can be even unknown (e.g., a goal in a football match, etc.) or dynamically triggered by users (e.g., shared service control, interactive instant messaging, a TV quiz show, in-game actions, etc.). In this case, the use of EED RTCP Feedback for IDMS implies the triggering of content- or action-based adjustments. Accordingly, a synchronous link between the application layer (i.e., operator generated events) and the transport/control layer (i.e., RTP/RTCP) is needed to align them in terms of timestamps. This is not a severe issue, since the Sync Manager will be co-located with the Media Server most of the times.

In case of a high frequency of events, setting an offset value for the RTCP report interval, by means of using the *trr-int* attribute, can help to save RTCP bandwidth (by restraining the transmission of too frequent Regular RTCP packets) while being able to use the (saved) bandwidth when events occur. This situation, however, is not considered in this paper (left for future study).

A similar mechanism exists in HBBTV[5]. It consists of inserting *"do it now"* events as elementary streams into the MPEG-TS to allow sync of dynamic events from extra applications (e.g., a question in an interactive quiz TV show, time-sensitive subtitles, etc.) with the live DVB content. However, the proposed EED RTCP Feedback is not only valid to dynamically trigger local inter-stream sync (even though this is not tested in this work), but also to enforce global IDMS adjustments in all the participants.

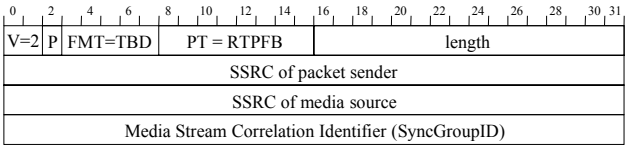
4.3 Rapid (Re-)Sync Request

If the initial compound RTCP packet (including SR, SDES and IDMS Settings) is lost, a Sync Client will not be able to synchronize the media playout until the next RTCP packet can be sent. This is undesirable. RFC 6051 [18] defines a new RTP/AVPF transport layer feedback message [17] to request the generation of an Early RTCP SR, allowing rapid inter-stream (re-)sync. A similar mechanism is proposed in this paper to be applied for IDMS purposes. A new RTP/AVPF transport layer feedback message [17], called RTCP-IDMS-REQ, is introduced to request the rapid generation (and transmission) of an RTCP IDMS Settings packet from the Sync Manager (see Fig. 5). The Payload Type (PT) of this RTCP message should be 205 [17], the Frame Message Type should be assigned by IANA (*Internet Assigned Numbers Authority*), and its length must be equal to 3. The *SSRC of the packet sender* field must indicate the Sync Client that is unable to synchronize, while the *SSRC of the media source* field must indicate the source of the media stream that the Sync Client is unable to synchronize. In contrast to the RTCP-SR-REQ [18], in which the *Feedback Control Information* (FCI) part is kept empty, in the RTCP-IDMS-REQ it must carry the Sync Group Identifier to which the sender of this message belongs [15].

Once a new RTCP-IDMS-REQ is received by the Sync Manager, it must generate an Early RTCP IDMS Settings packet. This mechanism can also be employed if a Sync Client has not received IDMS Settings in a (configurable) long time interval. Even though this mechanism is similar to the one in [18] to request rapid SRs, it is especially necessary since, in most implementations (e.g., [10, 11]), the IDMS Setting packets are only sent when the detected asynchrony exceeds an allowed threshold, and not regularly in each RTCP report interval as SRs.

4.4 Reduction of Channel Change Delays

The support for and rapid accommodation of latecomers are key issues to enable dynamic IDMS sessions. This is another useful applicability of the proposed RTCP-IDMS-REQ message. Once a latecomer joins an IDMS-enabled session, it must send an RTCP-IDMS-REQ message to the Sync Manager, which must send an Early RTCP IDMS Settings packet to rapidly bring the latecomer up-to-date. Upon receiving the IDMS Settings packet, the latecomer has the necessary sync info to start playing out the media stream in a time synchronized way with the other Sync Clients (thus preventing from either long annoying startup delays or initial playout inconsistencies). The timing diagram for the RTCP exchange processes in this SMS-based IDMS solution is illustrated in Fig. 6. It can be seen that, using EED RTCP Feedback, the IDMS latency for latecomers (i.e., the time interval between joining and acquiring IDMS) can be significantly reduced mainly due to the fact that the Sync Manager delay (Δt_2 in Fig. 6) is minimized. Two additional mechanisms could contribute to further reduce the latency for receiving the IDMS info (see Fig. 6). The first one consists of employing priority mechanisms for the transport of RTCP messages, e.g., by adopting a Differentiated Services (DiffServ) policy, as in [24]. This would help to decrease the Round Trip Time (RTT) delays and the loss probability for RTCP packets (out of the scope of this paper).



(TBD: To Be Determined)

Figure 5. RTCP IDMS-REQ Feedback Message.

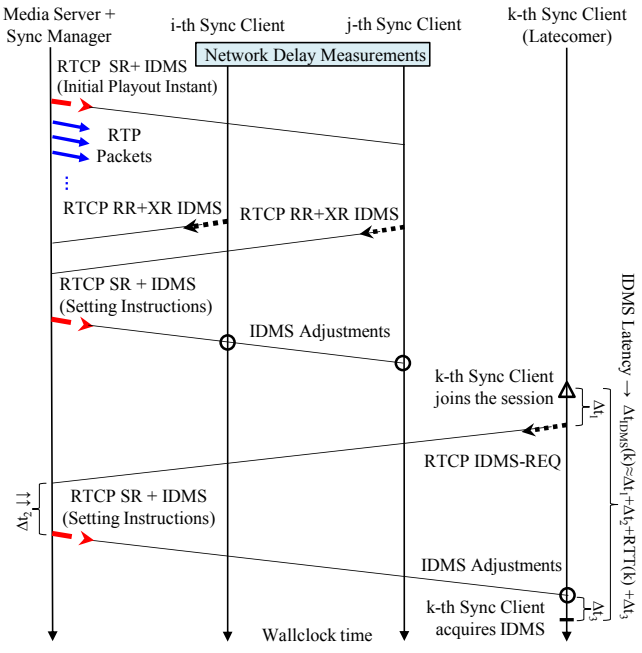


Figure 6. RTCP Message Exchanges for IDMS using SMS.

[5] Hybrid Broadcast Broadband (HBB) TV, http://www.hbbtv.org/.

The second one is based on the transmission of Early RTCP-IDMS-REQ messages by latecomers upon joining the session. According to [18], the delay since joining and sending an RTCP-IDMS-REQ message (Δt_1 in Fig. 6) should not be reduced to avoid flooding of requests at specific time instants (e.g., at the time a broadcasted sport event begins). While in this paper we adhere to this standard compliant rule, we will analyze in future research if this flash crowd effect is a real limiting issue in different large-scale SSM scenarios (e.g., networked quiz shows, gaming, IPTV, etc.). Our initial assumption is that the upstream bandwidth availability by the Sync Clients (which is not used for other purposes) and the aggregation and re-distribution mechanisms by Feedback Targets [19] do not entail a real constraint for allowing the transmission of Early RTCP-IDMS-REQ by Sync Clients. Moreover, it is assumed in [18] that all Sync Clients switch channels simultaneously, but even though using automated procedures (e.g., through notifications via the Electronic Program Guides in IPTV), this would not be a matter of a few seconds, but of minutes.

5. EVALUATION

Modeling and simulations were conducted using NS-2. The EED RTCP Feedback for IDMS was tested in a multicast scenario (Fig. 7) with four distributed Sync Clients belonging to the same logical group (Group 1 or G1) [11, 15], and with variable delays, and therefore Round Trip Times (RTTs), to the Media Server (see Table 1). All the links were bidirectional, their propagation delays were set to 10 ms, and their capacity was configured as shown in the figure. The Sync Manager was co-located with the Media Server which transmitted with a specific rate of θ=25 Media Units (i.e., video frames) per second (MU/s). Apart from the RTP/RTCP traffic, heavy and fluctuating background traffic (different cross-traffic flows following FTP/TCP, Pareto/UDP, CBR/UDP patterns) was configured over the network topology in order to cause significant different network jitter for each Sync Client, by forcing full usage of the links' capacities at some instants during the simulations. The scheduler clock of the simulator was used as a reference wall-clock in all the involved sync entities, thus ensuring the availability of a common clock source in all of them (as achieved by using a clock sync mechanism [16], such as NTP, in real setups). Additionally, significant playout rate deviations (skews and drifts) [11] were set (see Table 1) in order to force higher asynchronies between the Sync Clients, and to test if they could be successfully handled by our IDMS solutions[6].

In all the simulation tests, smooth playout adjustments were employed to acquire IDMS since, as it was shown in [11], this reactive technique outperforms the aggressive playout adjustments (*skips & pauses*) policy, because of the ability of achieving a more fine-grained sync, while minimizing the occurrence of long-term (annoying) playout disruptions.

5.1 Interactivity Comparison between Regular and EED RTCP Feedback

Figure 8 illustrates the playout (e2e) delay evolution for 3 Sync Clients to acquire IDMS when using EED RTCP Feedback, by employing the "sync to the mean playout point" policy [11]. First, it can be observed that all the Sync Clients were perfectly synchronized at the initial playout instant, despite of jitter and the e2e delay variability between them, because of the reception of an Immediate Initial IDMS Settings packet (Section 4.1). After that, it can be seen that the asynchrony between them progressively increased, mainly due to the configured deviations in their playout processes (Table 1). Every time an asynchrony exceeding τ_{max} (configured to 80 ms) was detected by the Sync Manager, it sent (multicast) an Early RTCP IDMS Settings packet to make the Sync Clients to get in sync with the selected IDMS reference (mean playout point among them [11]).

The same situation when using Regular RTCP Feedback was also simulated. The graphs for both cases were very similar. To clarify the differences among them, zoom views of the playout adjustment processes in each case are presented in Fig. 9. It can be observed that the asynchrony situation was corrected later when using Regular RTCP (left graph) than when using EED RTCP (right graph). This way, this figure shows that the IDMS latency was reduced when using EED RTCP Feedback, because of the minimization of the Sync Manager delay. As can be observed in both graphs, the playout adjustments did not start simultaneously in all the Sync Clients (due to the variable network delays to the Media Server, as shown in Table 1), but all of them finished their adjustments almost simultaneously at the target playout point included in the IDMS Setting packet (high performance in terms of coherence of SMS [1]).

In our simulated scenario, the Sync Manager was 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*. Assuming an approximate value of $avg(RTCP_{size})\approx1000$ bits (including the IDMS messages, plus transport and network layer headers, e.g. UDP and IP), and according to formulas in Fig. 1, a delay of up to 0.6 seconds 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 an IDMS Settings packet. This gives the maximum Sync Manager delay because of the bounded RTCP reporting rules. Even though the maximum playout asynchrony may slightly increase during this additional Sync Manager delay, the issue here is that, when using Regular RTCP Feedback, the out of sync situation will not be corrected during this time interval.

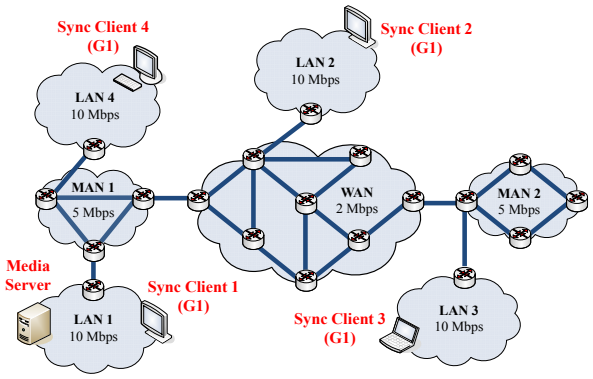


Figure 7. Simulated Scenario.

Table 1. Sync Clients' Parameters

Sync Client	Mean RTT	Rate Skew	Rate Drift
SC1 (LAN1)	~10 ms	γ_1 = 0.05 %	ε_1 = 0.02 %
SC2 (LAN2)	~125 ms	γ_2 = -0.02 %	ε_1 = 0.02 %
SC3 (LAN3)	~288 ms	γ_3 = -0.05 %	ε_3 = 0.02 %
SC4 (LAN4)	~125 ms	γ_4 = 0.015 %	ε_4 = 0.02 %

[6] The effect of the playout rate imperfections over the local and global media sync (especially IDMS) can be found in [11].

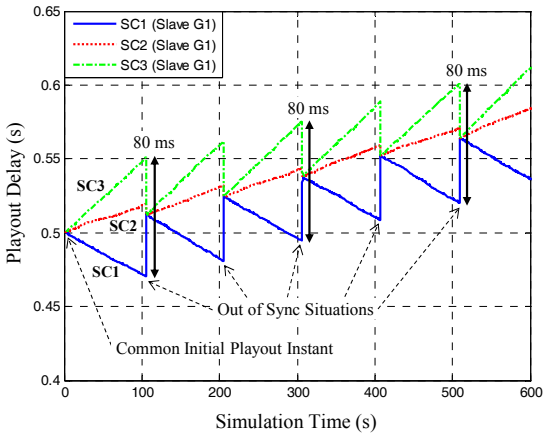


Figure 8. Playout (e2e) Delay Evolution when using EED RTCP Feedback for IDMS.

Note that, even without considering either the $T^{3550}_{RTCP\_d\_min}$ or the *trr-int* attribute, the differences between Regular and EED RTCP Feedback in terms of interactivity are significant, especially for those IDMS use cases with stringent sync requirements [1]. These differences would have been significantly larger (up to 5 s or even more, according to the Eqs. in Fig. 1 [13]) if any of the above mechanisms ($T^{3550}_{RTCP\_d\_min}$ or *trr-int*) were adopted due to the following two reasons. The first one is because of the larger Sync Manager delays. The second one is because of the larger delays needed to gather the IDMS reports from all Sync Clients, since their RTCP report interval would be lower in such a case. Therefore, asynchrony situations would be detected later.

Also, it is important to emphasize that these delay differences occur when using any of the policies for choosing a master IDMS reference presented in [11], although only the evaluation of one of them (sync to the mean playout point) is included in this paper due to space limitations.

Moreover, according to the Eqs. in Fig. 1, such delay differences would be much larger if the Sync Manager functionality would have been implemented as a part of an RTP receiver in a large-scale session (i.e., involving lots of Sync Clients), as can be inferred from the value of the RTCP report interval in Fig. 10, or if there were multiple senders in the session. Therefore, the proposed EED RTCP Feedback for IDMS would be even more beneficial if the Sync Manager were implemented as a part of a Sync Client (left for further study), provided that multicast feedback capabilities were available to that Sync Client.

To corroborate the benefits of using EED RTCP Feedback for IDMS, the fraction of MUs that were played out in all Sync Clients with an asynchrony larger than the allowed threshold was assessed, for different threshold values, when using both Regular and EED RTCP Feedback. Figure 11 shows the average results of 10 simulation runs (with different seeds for the random variables in each iteration). The following asynchrony threshold values were employed: sub-frame accuracy ($1/(2\cdot\theta)$=20 ms), frame accuracy ($1/\theta$=40 ms), 2 frames accuracy ($2/\theta$=80 ms) and 4 frames accuracy ($4/\theta$=160 ms).

It can be appreciated that the fraction of out of sync MUs when using EED RTCP Feedback is not as dependent on the allowed threshold as when Regular RTCP Feedback is used, since a less steep slope can be observed in the graph. Indeed, we can see that despite the fact that the differences are not very high for the upper threshold (160 ms), they become relevant when the threshold is reduced. For example, the fraction of out of sync MUs is almost double for a frame accurate threshold (40 ms), whilst it is more than double for the lowest threshold, comparing Regular to EED RTCP Feedback. This corroborates the better performance in terms of interactivity, because of the earlier correction of out of sync situations when using EED RTCP Feedback for IDMS. This is especially relevant for the IDMS use cases with stringent sync requirements [1]. Also, it is important to mention that although the percentages of out of sync MUs seem quite high, this fact does not mean that those asynchrony situations are annoying to human perception, because it is sufficient with setting an allowed threshold slightly lower than the normally noticeable asynchrony limits. For example, the maximum asynchrony value when using Regular RTCP Feedback for τ_{max}=80 ms in all simulations was 82.3 ms (higher than when using the EED RTCP), which confirms this assumption.

5.2 Fine-Sync for Media-Related Events

Figure 12 illustrates the same situation as in Fig. 8 when media-related events were triggered by the Media Server (through the Sync Manager) with a frequency of one event per 150 s (although they could be triggered dynamically or sporadically). In such a case, the Sync Manager sent IDMS Settings packets both as a response to the detection of out of sync situations (occurring in a non-deterministic way) and to the occurrence of media-related events (e.g., from application-dependent actions, such as post-advertisements, start of a football match, a penalty shot, etc.), despite that the asynchrony at that moment was lower than the allowed threshold. Table 2 shows the sync granularity with which those events were presented in the involved Sync Clients when using both Regular and EED RTCP Feedback (mean value and standard deviation of 10 simulation runs). It can be seen that the asynchrony for media-related events can range from a perfect sync (i.e., no delay differences) to the allowed threshold (or a slightly superior value) when using Regular RTCP, because there is no provisioning for syncing dynamic events. The asynchrony values from Table 2 for Regular RTCP can be checked through the graphical representation of the playout delays in Fig. 8. In contrast, it can be seen that those media-related events were presented with highly accurate sync levels when using EED RTCP Feedback. The obtained sync granularity is not perfect (i.e., asynchrony equal to zero), as expected, due to the configured playout rate deviations (Table 1).

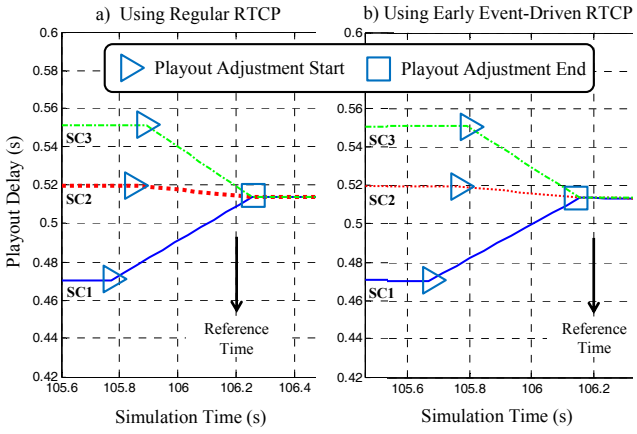


Figure 9. Zoom View of the Playout Adjustments to Achieve IDMS: Regular vs EED RTCP Feedback.

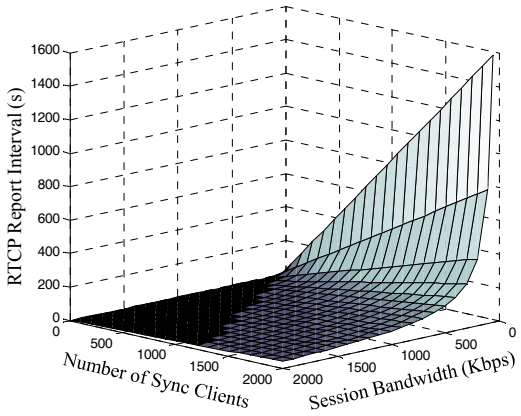


Figure 10. RTCP Report Interval for Sync Clients.

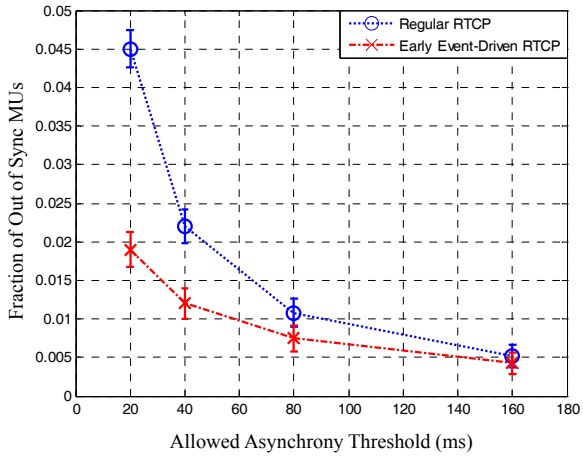


Figure 11. Interactivity Comparison between Regular and EED RTCP Feedback for IDMS.

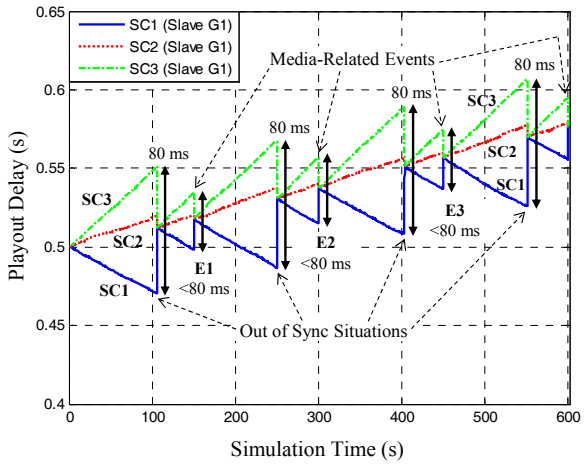


Figure 12. Playout (e2e) Delay Evolution when using EED RTCP for IDMS (with Media-Related Events).

Table 2. Sync Accuracy for Media-Related Events

Event (s)	ASYNCHRONY (ms)	
	Regular RTCP	EED RTCP
E1 (150)	38.5 ± 0.627	0.080 ± 0.022
E2 (300)	77.0 ± 0.804	0.073 ± 0.017
E3 (450)	29.5 ± 0.832	0.066 ± 0.019
E4 (600)	70.7 ± 0.799	0.071 ± 0.018

5.3 Rapid Accommodation of Latecomers

Figure 13 illustrates the same situation as in Fig. 8 when a latecomer (SC4) joined the session in progress at second 60. At that moment, SC4 sent an RTCP-IDMS-REQ message, and started buffering the incoming RTP packets. Once the Sync Manager received the RTCP-IDMS-REQ message, it sent an Early IDMS Settings packet to allow SC4 to become synchronized as soon as possible. Once SC4 received the IDMS Settings packet, it scheduled its playout controller to be able to synchronize (assuming SC4 is already able to start consuming the media stream because of the adoption of some of the techniques in [22-24]). Depending on the target playout point included in the IDMS Settings packet, it could be possible that some buffered media data need to be discarded by the latecomer when starting its playout process. In the simulated cases, SC4 experienced a maximum IDMS latency of 2.1 s (Δt_1 = 0.8 s, RTT = 0.125 s, Δt_2 = 0.6 s, Δt_3 = 0.575 s, see Fig. 6) when using Regular RTCP Feedback, whilst the one when using EED RTCP Feedback was decreased to 1.5 s (i.e., 2.1 – 0.6), mainly due to the fact that the Sync Manager delay (Δt_2 in Fig. 6) was minimized. Therefore, as discussed earlier, the use of EED RTCP Feedback can also significantly contribute to decrease the zapping delays in those media sessions requiring IDMS. Further research will be addressed to analyze the feasibility of reducing the other sources of delay (i.e., RTT, Δt_1 and Δt_3) when zapping in IDMS-enabled sessions. The first two (RTT and Δt_1) have been discussed in Section 4.3, whilst the third one (Δt_3) implies optimizing both the operation of the Sync Manager (IDMS target playout point calculation) and of the latecomer (buffering techniques and/or retrospective interpretation of the IDMS Setting packets).

5.4 Traffic Overhead

The same amount of RTCP packets were sent by the Sync Manager (co-located with the Media Server) during the 10-minutes session (around 1300-1320 packets, depending on the initial seed in each simulation) when using both Regular and EED RTCP Feedback, always adhering to the allowed RTCP traffic bounds [13]. Note that any differences were only originated at the initial playout instant and after detecting an event (e.g., an out-of-sync situation or a latecomer joins). In these cases, an Early IDMS Settings packet was sent using EED RTCP Feedback, but the next Regular RTCP transmission time was skipped (see Fig. 4). Therefore, the total number of sent RTCP packets did not differ.

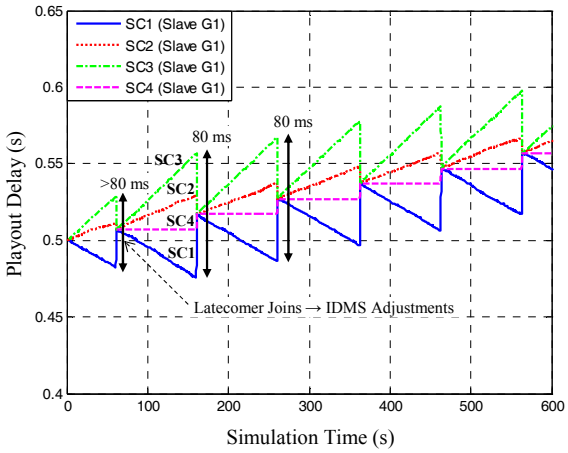


Figure 13. Rapid Accommodation of Latecomer (SC4).

6. CONCLUSIONS

In this paper, we have presented novel EED RTCP reporting mechanisms that enable higher interactivity (i.e., lower IDMS latency), flexibility, dynamism and accuracy when using a standardized RTP/RTCP-based solution for IDMS [15]. The simulation results provide evidence of the ability of the designed EED RTCP Feedback to achieve faster reaction to specific situations (e.g., an out of sync, or channel change delays) in IDMS-enabled sessions, as well as a finer granularity for syncing dynamic application-to-media events in all Sync Clients, compared to using Regular RTCP Feedback, while still adhering to the RTCP traffic bounds [13].

As future work, we plan to evaluate the benefits of the EED RTCP Feedback for IDMS by implementing a prototype in a real media framework (GStreamer). This will enable us 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. Additional research will be focused on analyzing and optimizing the different sources of delay to further decrease the IDMS latency when zapping (see Fig.6). Moreover, as can be inferred from Fig. 10, future research will also be needed to enable a really scalable IDMS solution involving lots of Sync Clients, such as Internet Radio or IPTV distribution channels. Otherwise, the RTCP report interval for the Sync Clients will increase up to the point that their IDMS reports will be sent with a very low frequency, thus probably providing outdated and unusable IDMS statistics.

7. ACKNOWLEDGMENTS

This work has been financed, partially, by Universitat Politècnica de València (UPV) under its R&D Support Program in PAID-01-10 & PAID-00-12 Projects, and by FP7/2007-2013 under grant agreement no. ICT-2011-7-287848 (HBB-NEXT project). We also want to thank Pablo Cesar and the paper's shepherd, Hermann Hellwagner, for their helpful comments.

8. REFERENCES

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

[2] Vaishnavi, I., et al., From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization, *Signal Processing: Image Communication*, 26(7), 370-377, August 2011.

[3] Mekuria, R., Cesar, P., Bulterman, D., Digital TV: the effect of delay when watching football, *EuroiTV '12*. Berlin (Germany), July 2012.

[4] ITU-T G.1050. Network model for evaluating multimedia transmission performance over Internet Protocol, 2007.

[5] van Deventer, O., et al., Advanced Interactive Television Service Require Synchronization, *IWSSIP'08*, Bratislava, June 2008

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

[7] Geerts, D., et al., Are we in sync? Synchronization requirements for watching online video together, *CHI 2011*, New York (USA), May 2011.

[8] Mauve, M., et al., Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications, *IEEE Transactions on Multimedia*, 6(1), February 2004.

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

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

[11] Montagud, M., Boronat, F., Enhanced adaptive RTCP-based Inter-Destination Multimedia Synchronization approach for distributed applications, *Computer Networks Journal*, 56(12), 2912-2933, August 2012.

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

[13] Schulzrinne, H., et al., RTP: A Transport Protocol for Real-Time Applications, *RFC 3550*, July 2003.

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

[15] van Brandenburg, R., et al., "RTCP for inter-destination media synchronization", draft-ietf-avtcore-idms-08, IETF Internet Draft, January 2013.

[16] Williams, A., et al., RTP Clock Source Signalling, draft-williams-avtcore-clksrc-02, IETF Internet Draft, Feb. 2013.

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

[18] Perkins, C., Schierl, T., Rapid Synchronization of RTP Flows, RFC 6051, November 2010.

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

[20] Kropfberger, M., Hellwagner, H., Evaluation of RTP immediate feedback and retransmission extensions, *ICME 2004*, 1751-1754, Taipei (Taiwan), June 2004.

[21] Liu, C., et al., RTP/AVPF compliant feedback for error resilient video coding in conversational applications, *ISCIT 2009*, 218-223, Incheon (Korea), September 2009.

[22] Ramos, F., et al., Reducing channel change delay in IPTV by predictive pre-joining of TV channels, *Signal Processing: Image Communication, 26(7)*, 400-412, August 2011.

[23] Begen A. C., Glazebrook N., Ver Steeg W., Reducing Channel-Change Times with the Real-Time Transport Protocol, *IEEE Int. Computing*, 13(3), 40-47, June 2009.

[24] Begic Z., Kos, M., Rapid synchronization of RTP multicast sessions, *International Journal of Computer Science and Network Security*, 10(9), pp. 42-47, September 2010.

[25] Fleury, C., Duval, T., Gouranton, V., Arnaldi, B., Architectures and Mechanisms to Efficiently Maintain Consistency in Collaborative Virtual Environments, *SEARIS 2010*, Massachussets (USA), March 2010.