Intrastream Synchronization for Continuous Media Streams: A Survey of Playout Schedulers

Nikolaos Laoutaris and Ioannis Stavrakakis
Department of Informatics & Telecommunications,
University of Athens, 15784 Athens, Greece
{laoutaris,istavrak}@di.uoa.gr

Abstract— **The transmission of real-time streams over best-effort networks has been an interesting research area for over a decade. An important objective of the research community has been to devise methods that cope with the variations of the network delay – also called delay jitter – that are an inherent characteristic of best-effort networks. Jitter destroys the temporal relationships between periodically transmitted media units (MUs) that constitute a real-time media stream, thus hindering the comprehension of the stream. Playout adaptation algorithms undertake the labor of the temporal reconstruction of the stream, which is sometimes referred to as the restoration of its intrastream synchronization quality. This paper surveys the work in the area of playout adaptation, aiming to concisely organize ideas that have been presented in isolation and identify the main points of differentiation amongst different schemes. The survey discusses issues related to the timing information, the handling of late media units, the quality evaluation metrics, and the adaptation to changing delay conditions.**

I. Introduction

Continuous media are characterized by well defined temporal relationships between subsequent media units (MUs). Information is only conveyed when these temporal relationships are preserved at presentation time (if altered during the transportation they will need to be reconstructed prior to presentation). The reconstruction of temporal relationships between media units of the same stream is referred to as *intrastream synchronization*. For video presentations, the temporal relationship refers to the spacing between subsequent frames, which is dictated by the frame production rate, typically 25 or 30 frames/second. For packet audio, the basic media unit is a voice sample, and the spacing between voice samples is determined by the sampling process. Temporal relationships also exist between MUs that belong to different streams, when these streams are to be consumed concurrently, as in an orchestrated audio-visual presentation (the lip-synchronization problem). The problem of synchronization between different, but related streams, is called *interstream synchronization* and it is out of the scope of this paper. For intermedia synchronization issues, the reader is referred to [1], [2], [3], [4].

A packet media receiver consists of a playout buffer for the temporary storage of incoming MUs and a playout scheduler for the presentation of MUs. The role of the scheduler is to provide a presentation schedule that resembles as much as possible the temporal relationships that were created by the encoding process. In doing so, the scheduler employs MU-buffering, the extent of which is bounded by the maximum end-to-end delay tolerance of the application. Bidirectional applications such as desktop video conferencing place very strict latency requirements, typically a few hundreds of milliseconds. On the other hand, unidirectional applications, for example video on demand (VOD), allow for much larger latencies which range from around one second, for responsive web-based distribution of short video clips, to several minutes, in near video on demand systems. All the proposed schemes provide for some compromise between the intrastream synchronization quality and the increase of end-to-end delay due to the buffering of media units. At the two extremes of this continuum of choices we have the buffer-less scheduler, that provides for the minimal stream delay by presenting frames as soon as they arrive, and the assured synchronization method, that completely eliminates the effects of jitter at the expense of a large stream delay.

In what follows we attempt to provide a structured presentation of proposed playout schedulers by examining the way they tackle the fundamental tradeoff between the synchronization quality and the imposed delay. Alongside the operational comparison of different schemes, an effort is made to indicate their suitability for different real world applications.

The remainder of the paper is organized as follows. Some background material and an outline are presented in Sect. II. Sect. III discusses the appropriateness of the various schemes for different media types. Sect. IV presents the family of time-oriented playout schedulers. Time unaware systems are covered in Sect. V. Sect. VI provides a comparison of different systems. Sect. VII briefly examines forward error correction techniques as well as multimedia caching and their relevance to playout adaptation. Sect. VIII concludes the paper.

II. Issues of Interest and Outline

For the assessment of intrastream synchronization quality several metrics have been proposed. They can roughly be categorized as being of first or second order (see Table II). First order metrics quantify “how much” synchronization loss occurs (e.g., expected frequency of gaps, accumulated duration of gaps, etc.). In most earlier schedulers, designed for packet voice systems, the metric of intrastream synchronization quality is the probability (or frequency) of a voice packet being discarded as a consequence of a ”late” arrival due to a large network transfer delay. In buffer-oriented schedulers (Sect. V) the continuity quality of the stream is affected by underflow discontinuities

This work and its dissemination efforts have been supported in part by the IST Program of the European Union under contract IST-1999-10160 (VideoGateway).

(when the buffer empties) and overflow discontinuities (when an arriving MU finds the playout buffer full) – so the synchronization metric must cater to both events. Also, in systems that modify the playout delay by affecting the duration of MUs (Sect. V-C), the metric must also consider the discontinuity introduced by the system itself (e.g., due to the expansion (or reduction) of the presentation duration of a video frame).

Second order metrics capture the appearance pattern of synchronization loss occurrences. The human perceptual system is known to be more sensitive to a small frequency of long-lasting disruptions than to a higher frequency of short-lived disruptions [5]. This is due to the human perceptual inability to notice small deviations of presentation rate. As a result, a better perceptual quality can be expected by replacing large continuity disruptions (underflows and overflows) with shorter ones (e.g., truncated or extended MU presentation durations). A hybrid metric that involves both first and second order features has also been proposed [6]. Despite the plethora of proposed metrics, only a few experimental perceptual studies have been conducted for the assessment of acceptable values for these metrics [7], [8], [9], especially for the more advanced metrics, and under different media encoding methods.

Throughout the rest of the paper various playout schedulers from the literature will be presented. One main point of differentiation will be whether the systems use or do not use, timing information. Time-oriented schedulers (Sect. IV) put timestamps on MUs and use clocks at the sender and the receiver in order to measure the network delay or the differential network delay (jitter); see Table I for an overview of time-oriented schedulers. Buffer oriented schedulers (Sect. V) implicitly assess the current level of network jitter by observing the occupancy of the playout buffer; see Table II for an overview of buffer-oriented schedulers. In any case, the level of synchronization between the sender and the receiver greatly affects the design and the capabilities of the system. A general classification of playout schedulers is given schematically in Fig. 1.

Systems that have some protocol for the synchronization of their clocks are said to have a *global clock*. The existence of a global clock allows for the exact measurement of the network transfer delay of a MU which, along with the buffering delay at the receiver, makes up the total end-to-end delay of the MU. With such knowledge, the receiver can then guarantee that a MU be delivered before an available (requested) end-to-end delay budget is exhausted.

When the network delay is unknown, no guarantee in absolute values can be provided for the interactivity of the system (as required by some demanding bidirectional applications). Differential delay methods (Sect. IV-C) do not require a global clock and, thus, cannot measure precisely the network delay of a MU as there might be an offset between the two clocks. These methods do not consider absolute delays but rather operate on delay variations (captured by the difference between subsequent delay measurements, thus eliminating the offset). They strive to maintain a fixed tradeoff between the perceived delay and the synchronization quality of the stream, across time varying jitter.

Schedulers with approximated clock synchronization (Sect. IV-D) try to bound the offset (thus the uncertainty) between the two clocks. This is achieved by using the *virtual clock* algorithm

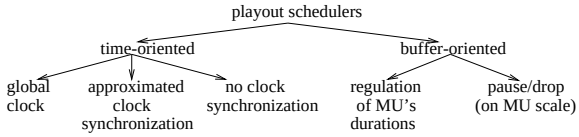

Fig. 1. General classification of playout schedulers.

under which the clock of the receiver adopts as local time the timestamp from a reference packet (sometimes called a probe) which is sent by the sender. The consequence of this virtual synchronization of the receiver clock to the sender clock is that within the context of this communication the clock of the receiver will be delayed by $t_{s\to r}$ time units compared to the clock of the sender. $t_{s\to r}$ denotes the network delay of the reference packet. The receiver forwards the reference packet back to the sender (incurring a delay $t_{r\to s}$), thus allowing for the exact measurement of the round-trip-time (RTT) between the two end-points. The time offset between the two clocks is then securely identified as a value in the interval $[0 \ldots \text{RTT}]$ [1]. See [10], [11] for more issues related to delay estimation and clock synchronization.

Nearly all presented systems have mechanisms to detect increases in the end-to-end latency: time-oriented systems can tell if a MU is "late" by comparing its arrival timestamp with its scheduled playout time; buffer oriented system realize a latency increase when they detect an over-built queue of MUs waiting to be displayed. The way late MUs are handled divides the proposed schemes into two categories: delay-preserving schemes, where all late frames are discarded to preserve the delay requirements of the stream; non delay-preserving schemes, where some (or all) late packets are accepted for presentation with the aim to protect the continuity of the stream from further degradation due to the discard of MUs that have been late.

The majority of the proposed schemes adjust the occupancy of the playout buffer dynamically, in response to varying network delay jitter. The playout buffer is increased to compensate for increased delay variability, protecting the continuity of the stream; it is decreased in times of reduced jitter, to provide for a similar synchronization quality but with a reduced stream latency. Packet audio systems with silence detection modify the playout buffer by taking advantage of silence periods, which they use to make adjustments on a per-talkspurt basis. Packet video receivers usually drop a frame to reduce latency, or stop the presentation of frames for one frame period (a pause); we refer to this method as the pause/drop method. More advanced systems (Sect. V-C) change the occupancy of the playout buffer by regulating the duration of video frames; they present frames faster (instead of dropping) to reduce latency, and they present frames slower to avoid underflows. It has been realized that this regulation approach has some advantages, as slight modifications of presentation rate may be unnoticed due to human perceptual limitations, and thus, the adjustments are better concealed.

[1] When the clock at the sender reads t, the clock at the receiver will be at $t - t_{s\to r}$. The offset will be 0 if $t_{s\to r} = 0$, meaning that the entire RTT was due to the network delay from the receiver to the sender. At the worse case the offset will be equal to RTT (meaning that $t_{s\to r} = \text{RTT}$ and $t_{r\to s} = 0$).

III. Media type and playout adaptation

In the following sections, playout schemes will be taxonomized according to the precision of timing information. This section provides a parallel organization based on the applicability of schemes for different media types.

Streaming media can be either *continuous* or *semi-continuous*. The first category includes media streams with a regular inter-MU interval that is maintained throughout the duration of the presentation. Typical examples are streaming video (live or stored) and streaming audio (e.g., web-radio). Semi-continuous media are characterized by inactivity periods that intervene between the periods of continuous MU flow. Spoken voice with silence detection is a widely used semi-continuous medium. Video programs, where the user watches a short video clip and then selects the next one (e.g., news repository consisting of sequences of short clips) also belong to the same category.

The main difference between these two media types is that the inactivity periods of semi-continuous media give the playout scheduler the opportunity to adjust the playout point for the imminent activity period without affecting its continuity. The playout schedulers of sections IV-B to IV-D modify the silence period in an audio conversation and by doing so they adjust the playout point[2] for the next spoken phrase. Playout schedulers for continuous media do not have the "luxury" of inactivity periods, but have to act on inter-MU intervals to adjust the playout point. One may wonder why a scheduler should alter the temporal relationships of MUs when it's main job is to restore these same relationships. The answer to this question is that by intentionally harming the continuity of a stream under the proper conditions, the scheduler anticipates a future quality improvement. This anticipation stems from the improved ability to conceal long-lasting continuity disruptions. Such schemes – for continuous video – are presented in Section V which is devoted to buffer-oriented playout schedulers (see Table II for a quick summary). Although some of the ideas therein could also be applied to continuous audio, we are not aware of such applications. This may be attributed to the fact that audio is believed to be more sensitive than video to the manipulations introduced by a buffer-oriented scheduler.

Finally, we note that some schemes are equally appropriate to both continuous video and continuous audio. Such schemes are those that decide only on the initial amount of buffered data and do not intervene thereafter. Section IV-A presents two systems of this category. Also, the simplistic approach of not implementing a dejitter buffer (Section V-A) has been used in both audio and video applications (e.g., in the VIC Mbone tool).

IV. Time-oriented playout schedulers

This section presents some common time oriented playout schedulers, i.e., schedulers that timestamp MUs and use local or global clocks to determine the presentation instant and the duration of each MU.

[2]Here the playout point, i.e, the scheduled playout time for each voice sample, depends on the initial delay that is imposed on the entire talkspurt.

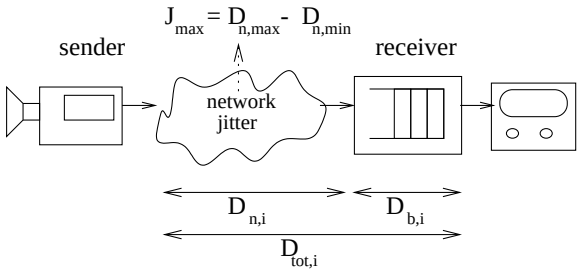

Fig. 2. The total end-to-end delay of the ith MU, $D_{tot,i}$, is composed of a variable network delay component, $D_{n,i}$, and a buffering delay component, $D_{b,i}$, at the playout buffer. The network delay ranges from a minimal value – only the propagation component $D_{n,min}$ – up to a maximum delay $D_{n,max}$. J_{max} denotes the maximum network delay variability.

A. The assured synchronization under bounded delay jitter

The ideal intrastream synchronization quality is achieved by completely eliminating any kind of distortion in the temporal relationships of MUs and completely restoring the stream to its initial form. This objective must be achieved on the fly as MUs arrive at the receiver, having crossed a network that alters the spacing between MUs by imposing a variable network transfer delay. The network transfer delay consists of two components: a static propagation delay and a variable queueing delay, due to variable waiting times in the queues of intermediate network nodes. If the delay variability is unbounded, meaning that an infinitely long interarrival period may appear, then no scheduler with a finite buffer can eliminate the discontinuities from the reconstruction process.

Under some service classes, e.g., in ATM CBR or in IETF's Guaranteed Service, the network is able to bound the maximum delay difference that can be experienced over a certain data path, and a total resynchronization at the receiver becomes feasible. We refer to this synchronization-optimal playout of frames as the *assured synchronization*. Two slightly different approaches have been proposed for the implementation of the assured synchronization. For their description, as well for use in the rest of the paper, the following notation is introduced (see also the delay diagram in Fig. 2): $D_{n,i}$ denotes the network delay of the ith MU; $D_{b,i}$ denotes the buffering delay of the ith MU; $D_{tot,i}$ denotes the total end-to-end delay of the ith MU (network and buffering parts); $D_{n,min}$ denotes the minimum network delay; $D_{n,max}$ denotes the maximum network delay; J_{max} denotes the maximum difference in any two network delays,i.e., $J_{max} = D_{n,max} - D_{n,min}$. A delay quantity without the index i refers to the entire stream and not to a particular media unit.

It has been shown that the ith MU causes a synchronization-loss event when its network delay, $D_{n,i}$, is larger over all previous network delays, $D_{n,j} : 1 \leq j < i$, plus the initial buffering delay for the first MU, $D_{b,1}$ (see [12], [13], [14], [15]). Thus, to ensure no loss of synchronization it suffices to guarantee that the first MU incurs a total delay $D_{tot,1} = D_{n,1} + D_{b,1}$ which is no less than $D_{n,max}$. The total delay of the stream is then equal to the total delay of the first MU, $D_{tot} = D_{tot,1}$. Depending on whether $D_{n,1}$ is known, the following two schemes provide for the assured synchronization by adding an appropriate $D_{b,1}$ to the first MU. The first scheme requires that J_{max} be known while the second scheme requires that $D_{n,max}$ be known.

A.1 Unknown $D_{n,1}$.

If $D_{n,1}$ is unknown (timestamps are not used), or if it can not be precisely measured (the two clocks are not synchronized), then to guarantee an assured synchronization, the first MU must be kept in the buffer for an interval equal to the maximum delay difference, i.e., $D_{b,1} = J_{max}$, before the presentation of frames is initiated by extracting frames from the head of the playout buffer (see Geyer et al. [16]). The initial buffering delay protects the synchronization of the stream against the worst possible scenario which corresponds to the first MU experiencing the minimum network delay, $D_{n,min}$, while a subsequent frame experiences the maximum network delay, $D_{n,max}$. The total end-to-end delay of the stream becomes: $D_{tot} = D_{tot,1} = D_{n,1}+J_{max}$, taking values in $[D_{n,min}+J_{max}, D_{n,max}+J_{max}]$, since $D_{n,min} \leq D_{n,1} \leq D_{n,max}$. In any case, the total end-to-end delay $D_{tot,1}$ of the first MU is no less than the maximum network delay $D_{n,max}$, so it is guaranteed that no packet will arrive late.

A.2 Known $D_{n,1}$.

The assured synchronization method can be implemented more efficiently by using timestamps and a global clock to accurately measure the delay of the first MU and add the minimum buffering delay that makes the total delay of the first MU exactly equal to $D_{n,max}$. This was not the case in the previous implementation that was unaware of $D_{n,1}$ and had to assume $D_{n,1} = D_{n,min}$ and require an additional delay J_{max} to ensure that $D_{tot,1} \geq D_{n,max}$. The improved implementation (see Baldi and Ofek [17]) achieves a potentially smaller end-to-end delay, while continuing to guarantee an absolute synchronization at the receiver. Assuming that the scheduler knows $D_{n,1}$, an uninterrupted presentation schedule can be provided by keeping the first MU in the playout buffer for an additional interval $D_{b,1}$, so that the total delay of the MU becomes: $D_{tot,1} = D_{n,1}+D_{b,1} = D_{n,max}$. This way, it is guaranteed that no MU will experience a larger delay, thus no loss of synchronization will occur. Note that in comparison to the first method, a delay reduction of $D_{n,1} + J_{max} - D_{n,max} = D_{n,1} - D_{n,min}$ has been achieved (this is the *excess resynchronization delay*); this gain can be as large as $D_{n,max} - D_{n,min} = J_{max}$ and it is equal to zero when the first MU has experienced the minimum network delay. Despite this reduction, the delay performance may still be inadequate for interactive applications due to the total stream delay $D_{tot} = D_{n,max}$ which may be in the order of seconds, while the average network delay, $D_{n,mean}$, and the acceptable D_{tot} may be in the order of milli-seconds.

B. Allowing for the latency/synchronization tradeoff by allowing for loss due to MU-lateness

The intrastream synchronization performance of the assured synchronization method can not be matched by any other method. Its weakness lies on its potentially poor delay performance that makes it inadequate for interactive applications. This is due to the fact that even the improved version (with a global clock), imposes a prohibiting end-to-end delay (equal to the maximum network transfer delay $D_{n,max}$). In most modern packet networks there is no guarantee of an upper bound on the delay and even when a bounded[3] $D_{n,max}$ exists it is seldomly experienced by a MU as reported by recent delay studies [18], [19], where it shown that the probability of a long delay (given by the area under the tail of a delay distribution, see Fig. 3 for an illustration) is very small. Clearly, delaying the entire stream for $D_{n,max}$, in order to guarantee the on-time arrival of even the rare slow MUs, leads to poor delay performance. Furthermore, modern video and audio codecs can tolerate a substantial amount of packet loss (1-5% for voice) with acceptable degradation of the perceived quality. Consequently, playout schedulers can sacrifice an amount of lost MUs (which have been dropped as being excessively late) in order to decrease the overall delay of the stream and effectively support real-time applications.

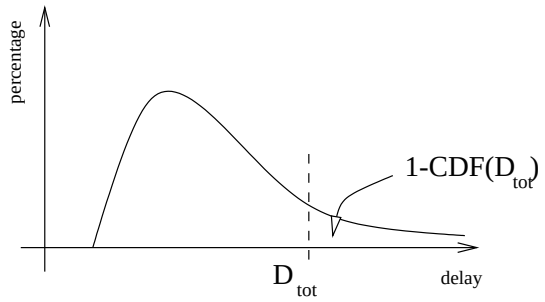

Fig. 3. A typical network delay distribution. Under *method-I* of Naylor and Kleinrock, all MUs with network delay larger than D_{tot} are considered lost. The percentage of lost packets, under this distribution is, $p_{loss} = 1 - CDF(D_{tot})$.

Schedulers destined for bidirectional interactive real-time applications usually try to provide a small, constant total end-to-end delay (D_{tot}) by utilizing precise timing information and allowing for packet lateness. Baldi and Ofek [17] have shown that given a small network delay, a system can be configured (packetization, compression, rendering) to provide a total delay as small as 100 msec. Packets that arrive at the receiver with a network delay larger than D_{tot} are discarded (delay-preserving schedulers or the *absolute delay method*). The choice of D_{tot} regulates the tradeoff between intrastream synchronization quality and delay. When the sender and receiver timestamps (used for the measurement of the network delay) come from synchronized clocks (GPS,NTP [20]), then it is guaranteed that any played MU will be delivered with an accurate D_{tot} (e.g. the Concord algorithm [14], [21]). The Concord algorithm assumes the existence of synchronized clocks at the sender and the receiver, and uses MU timestamps to construct a packet delay distribution (PDD) which is an estimate of the real distribution, possibly non-stationary, that packets face within the network. Having approximated the cumulative distribution function (CDF) of the PDD – i.e., the probability that the network delay of the ith MU is smaller than t: $P\{D_{n,i} \leq t\} = CDF(t)$ – one can choose a value for D_{tot} that provides for the desired compromise between packet loss and delay. The compromise is expressed as a function of the delay, which is D_{tot} for all MUs, and the packet loss probability, which is given by: p_{loss} =1−CDF(D_{tot}) (Fig. 3). By dynamically updating the

[3]For best-effort networks the maximum delay of a delivered packet corresponds to the case where it finds all queues of intermediate nodes nearly full, but with enough space to accept it.

PDD, the system provides a constant synchronization quality (p_{loss}) by adjusting the delay; D_{tot} is increased to smooth-out increased jitter, and it is decreased when jitter drops, to enhance interactivity.

C. Playout schedulers that do not require a global clock

A global clock can provide the utmost interactivity precision; it is the only mechanism that is capable of supporting the most stringent – in term of delay – policy, that demands that all MUs be presented at a constant (small) delay, or be discarded. A global clock is seldomly available, and thus, the majority of playout schedulers – those that do not require a constant end-to-end delay guarantee – operate on delay differences and not on absolute delays. When considering delay differences, the two clocks need not be synchronized, as their offset is cancelled when taking differences of timestamp values. The only requirement is that they run at approximately the same speed (they do not drift). The fundamental idea behind such systems is that the total delivery delay of MUs (encoding to presentation time) need not be constant, or confined under an absolute value, but it can fluctuate in response to changing network delay variability, so that a level of synchronization quality (e.g. percentage of late packets), or the more relaxed requirement of a constant tradeoff between continuity and delay, be maintained. Network delay differences are used as indications of the current jitter level, and drive the regulation of the playout buffer.

A pioneering work in this field is that of Naylor and Kleinrock [12]. It presents a playout scheduler that adaptively delays the first packet of a talkspurt based on recent jitter measurements. Specifically, the receiver logs the last m delays of MUs prior to the initiation of a new talkspurt and extracts the k partial range, $D(m,k)$, which is the maximum difference between the m samples having first discarded the k largest delays[4]. The first MU of the talkspurt is delayed for $D(m,k)$. The partial range is used to eliminate isolated cases of extremely large delay that do not have a significant impact on the probability of a late arrival. The tradeoff between gap probability and delay is controlled by the level of conservatism in the partial range, i.e., by the relative selection of m and k. As $D(m,k)$ is smaller than the maximum network delay, some packet lateness will occur. Two methods are proposed for the handling of late packets: *method-I*, preserves the delay of the stream by discarding late packets (*delay-preserving* method); *method-E*, expands the delay of the stream – in favor of continuity – by presenting late frames (*data-preserving* method).

A popular playout method for the adjustment of the playout buffer across time-varying network delay environments uses timestamps, without assuming global clock synchronization, to approximate the one way network delay $\hat{d}$ and its variability $\hat{v}$. The presentation time of the first MU of a talkspurt, p_i, (let it be the ith MU of the conversation) is scheduled for:

$$p_i = t_i + \hat{d}_i + \beta \cdot \hat{v}_i \tag{1}$$

t_i denotes the generation time of the ith MU according to the sender's clock. The variation coefficient β is used to set the playout time "far enough" beyond the delay estimate so that only an acceptable amount of packets will miss their playout time; in essence β regulates the synchronization/latency tradeoff ($\beta = 4$ in [22]). Ramjee et al. [22] propose four synchronization algorithms that differ only in the way they derive the estimate $\hat{d}_i$. The estimation is carried out on a per packet basis – using as input the network delay of the ith MU, d_i – while delay adjustments are applied on a per talkspurt basis. No clock synchronization is assumed, d_i is the difference between the arrival timestamp a_i and the generation timestamp t_i.

$$\begin{aligned} \hat{d}_i &= \alpha \cdot \hat{d}_{i-1} + (1-\alpha) \cdot d_i \\ \hat{v}_i &= \alpha \cdot \hat{v}_{i-1} + (1-\alpha) \cdot |\hat{d}_i - d_i| \end{aligned} \tag{2}$$

Two of the proposed algorithms are based on the linear recursive estimator of equation (2) ($1-\alpha$ is the *gain* of the filter, see [23] for details), the third algorithm is adopted from the NEVOT audio tool, and a fourth is a novel algorithm with delay spike detection capabilities and dual mode of operation, aiming at improving performance in delay environments with sharp delay spikes, as the ones reported in [24].

The last algorithm has been enhanced by Moon et al. [25] by substituting the linear recursive estimators of $\hat{d}_i$ and $\hat{v}_i$ with the calculation of a percentile point q of the underlying network delay distribution. This is done by logging the last w packet delays (no clock synchronization) and using their qth percentile point as the playout delay for the next talkspurt. By utilizing more detailed information about the delay distribution, the algorithm outperforms the initial algorithm by Ramjee et al. [22], and approaches an optimality bound, which is also derived in the same work.

D. Playout schedulers with approximated synchronization – Virtual Clocks

So far we have presented playout schedulers with global clock synchronization ([14], [21]) and without global clock synchronization ([12], [22], [25]). Schedulers with approximated clock synchronization fill the gap between the two extreme approaches. Such systems do not require a global clock, so they cannot guarantee a delivery delay in absolute values, but they provide a soft delivery guarantee that is more specific than the freely fluctuating delay of differential-delay systems, where the network delay component is completely unknown. A bound on the total delivery delay (D_{tot}) is established by measuring the round-trip-time (RTT) between the communicating end-points, and assuring that no MU will be presented with a delay that exceeds some expression that involves the RTT.

In Roccetti et al. [26], periodic probe packets and a three-way-handshake protocol are used for the exact measurement of the round-trip-time between the communicating end-points. The clock of the receiver is virtually synchronized to the clock of the sender by adopting as local time the timestamps of the probe packets. This immediately leads to a clock offset equal to the one-way network delay of the probe packet, t_0. A playout delay of t_0 would be too small, leading to increased packet lateness, so the clock of the receiver is delayed by an additional RTT, as measured by the latest probe packet, to give an overall time gap between the two clocks equal to t_0+RTT, i.e., the clock of the receiver falls t_0+RTT time units behind the clock

[4] $D(m,0)$ is the total range, the maximum delay difference out of the entire set of m delay samples

of the sender. Packets that arrive at the receiver with a timestamp larger than the local clock are buffered; packets that arrive with timestamps smaller than the local clock are considered too late and are discarded (their network delay has been larger than t_0+RTT); packets are extracted from the buffer and played when the local clock equals their timestamp. By refreshing the RTT every second, the algorithm regulates the playout point in accordance to the current network delay. Clock adjustments are enforced on the receiver only during silence periods to avoid time gaps caused by the adjustment of the clock during talkspurts.

A similar approach is followed by Alvarez-Cuevas et al. [27] with the use of probe packets, not periodically, but rather at the beginning of every silence period. Having measured the RTT, RTT/2 is used as an estimate of the one way delay of the talkspurt and is sent to the receiver with a reference packet. The receiver uses RTT/2 as the estimate of the network delay and adds an additional delay component with the aim of achieving a fixed target end-to-end delay D_{tot} that results in only 1% packet lateness. The additional delay up to D_{tot} should be $J_{max}/2$, where J_{max} denotes the maximum delay variability ($D_{n,max} - D_{n,min}$). However, $J_{max}/2$ is not known apriori, but is approximated and corrected by observing the extent of synchronization errors and increasing D_{tot} accordingly. By dynamically measuring RTT and adjusting D_{tot}, the algorithm adapts to network delay fluctuations and maintains the targeted synchronization quality. In the same work, a second method for the estimation of the target D_{tot} is described. It identifies the "fastest" packet – the one with the smallest delay (propagation only) – by looking for the packet that incurs the largest waiting time in the buffer; it is assumed that this is the fastest packet. D_{tot} is approximated as the network delay of the fastest packet plus the largest observed difference in buffering delays, which should approach J_{max}, resulting in approximately 1% packet lateness. Both methods can be improved by employing the method of hop-by-hop network delay accumulation which results in very accurate network delay estimations (only the local clocks of intermediate nodes are involved, each node accumulates its own added delay, see Montgomery [10]).

E. Non delay-preserving playout schedulers

A scheduler is characterized as being *delay-preserving* if it does not present late MUs, i.e, MUs that have missed their scheduled playing time. *Non delay-preserving* schedulers may accept and present a late MU, instead of discarding it, to protect the continuity of the stream from further degradation[5].

In delay-preserving playout schedulers, the arrival time of a MU, is bound to fall in one of the two possible regions: the *acceptance region* – limited by the targeted end-to-end delay – where the MU waits in the playout buffer for its playout time; and the *discard region*, for arrivals with a total delay longer than the targeted D_{tot} . The playout scheduler proposed by C. Liu et al. [2] violates the delay-preserving discipline of the two-region approach, by introducing the *no-wait region*, which lies between the other two regions (see Fig. 4); all arrivals in the no-wait region are played immediately. An arriving MU with delay that places it in the no-wait region is a MU which has missed its targeted D_{tot} , but not long enough to be discarded, so it is played immediately to prevent further degradation of synchronization caused by a media unit discard. With the acceptance and the presentation of a late MU, the end-to-end delay for subsequent MUs is increased. The boundary values that define the three regions are set by taking into consideration user-defined input-parameters which are: the maximum synchronization error, the maximum variance of synchronization error, and the maximum acceptable MU loss ratio. The scheduler monitors the percentage of MUs that fall in each region and adjusts their bounds so that the end-to-end delay is minimized and the user intrastream synchronization requirements are met. The scheduler does not offer an assured end-to-end delay as it does not use a global clock but only a virtual clock on the receiver.

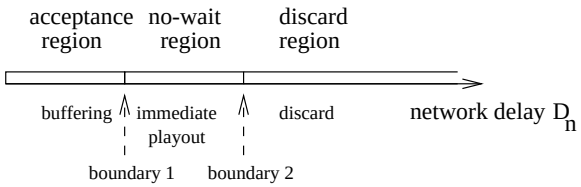

Fig. 4. Network delays fall in one of the three possible regions. MUs are buffered if they arrive early (D_n in the acceptance region), they are presented immediately if they are slightly "late" (D_n in the no-wait region), they are discarded if they arrive too late (D_n in the discard region).

The playout scheduler proposed by H. Liu et al. [28] is also non delay-preserving, as in some occasions it chooses to present a late video frame. The scheduler enters a synchronization recovery phase immediately following the arrival of a late packet. During that phase the scheduler determines the presentation duration for the late frame. A full presentation duration is undesirable as it increases the end-to-end delay of subsequent frames. On the other hand, a truncated presentation – up to the scheduled presentation instant of the next frame – might truncate the late frame excessively, causing motion jerkiness that is easily detected by the end user. The scheduler has a bounded minimum frame duration – such that motion jerkiness is not detectable – and can choose to apply it to a series of frames following the late arrival, thus progressively reducing the added delay due to the late arrival. This approach compresses the delay successfully, while at the same time protects the quality of intrastream synchronization by employing a rather "mild" delay control function. Another interesting feature of the scheduler is that it uses a second order continuity metric called RMSE. A user requested threshold RMSE is maintained by the scheduler across different transmission conditions by regulating the buffering delay accordingly.

V. Buffer-oriented playout schedulers

The family of buffer-oriented playout schedulers involves schedulers that deal with the fundamental synchronization/latency tradeoff but do not require the timestamping of MUs or the use of clocks. Buffer-oriented schedulers share some resemblance with time-oriented schedulers that use differential delay methods to adjust the playout point. Instead of using timestamps for the assessment of the current level of delay jitter, buffer-oriented schedulers implicitly assess jitter by observing the occupancy of the playout buffer. The adjustment

[5]The continuity of the stream has already been harmed by the late arrival.

author	media	global clock	delay performance	sync. performance	delay adaptation
assured synchr. (Geyer et al.) [16]	audio/video	not assumed	$D_{n,1} + J_{max}$	no loss of synch.	delay static during connection
improved assured synchr. (Baldi et al.) [17]	audio/video	assumed	$D_{n,max}$	no loss of synch.	delay static during connection
Concord alg. [14], [21]	audio/video	assumed	variable→min	static-guaranteed	based on PDD estimation
Naylor and Kleinrock [12]	audio+sil. det.	not assumed	stat. tradeoff	stat. tradeoff	per talkspurt (partial range filter)
Ramjee et al., [22]	audio+sil. det.	not assumed	stat. tradeoff	stat. tradeoff	per talkspurt (recursive filter)
Moon et al. [25]	audio+sil. det.	not assumed	stat. tradeoff	stat. tradeoff	per talkspurt (percentile point)
Roccetti et al. [26]	audio+sil. det.	VC, t_0+RTT	stat. tradeoff	stat. tradeoff	periodic (1 sec) (RTT based)
Alvarez-Cuevas et al. [27]	audio+sil. det.	VC, RTT/2	variable→min	1% MU lateness	per talkspurt (RTT based)
C. Liu et al. [2]	audio/video	VC	variable→min	satisfy user input	per MU (delay region based)
H. Liu et al. [28]	audio/video	VC	variable→min	satisfy user input	per MU↑, per W MUs ↓

TABLE I

OVERVIEW OF SURVEYED TIME-ORIENTED SCHEDULERS (TIMESTAMPING + SOME FORM OF CLOCK SYNC.). THE ABBREVIATION (VC) INDICATES A VIRTUAL CLOCK SYNCHRONIZATION METHOD, WITH SOME (USUALLY MENTIONED) APPROXIMATED OFFSET. THE LABEL "STAT. TRADEOFF" IDENTIFIES SYSTEMS WHERE A CONSTANT TRADEOFF BETWEEN SYNCHRONIZATION AND DELAY IS MAINTAIN ACROSS DIFFERENT LEVELS OF JITTER (EQUATION (1) GIVES SUCH AN EXAMPLE).

of the playout point is based on the occupancy of the buffer. The lack of timing information precludes any kind of absolute total end-to-end delay guarantees for the MU presentation epochs. The only "visible" delay component for the scheduler is the buffering delay of the MUs at the playout buffer. Various synchronization/latency objectives can be optimized by focusing on the tradeoff between media continuity and buffering delay. The total end-to-end delay, although unknown (and fluctuating), can be implicitly controlled as a consequence of the regulation of the buffering delay component; the suppression (expansion) of the buffering delay leads to the suppression (expansion) of the total end-to-end delay with a subsequent cost (gain) in intrastream synchronization quality. Delay performance that can approach the requirements of interactive applications is feasible, but cannot be guaranteed in absolute values. Due to this uncertainty concerning the interactivity of the system, buffer-oriented schedulers are usually employed in video applications where the interactivity requirements are more relaxed than in audio applications.

A. No initial buffering: Self-adjusting and Buffer-less schedulers

Presenting the first MU as soon as it arrives is a simple and rather successful scheme. It provides a potentially low initial delay (only the network part), and if no jitter occurs, it is a fairly good solution. In the common case in which jitter exists, the first underflow will occur as soon as some MU experiences a network delay that is greater than the delay of the first MU ($D_{n,1}$). The *self-adjusting* schedulers present all MUs that arrive at the receiver, thus, the playout buffer builds up as a natural effect of the induced underflows (since the mean arrival rate equals the mean presentation rate and an underflow is analogous to a "server vacation"). This results in a stream presentation with a small initial delay (no initial buffering) but also with an initially poor synchronization quality (frequent underflows); continuity is improving with time, as the dejitter buffer expands with underflows, but this also increases the perceived delay. To control the delay, some of the late MUs must be discarded. Contrary to time-oriented systems, a MU cannot be declared as late by means of comparison of its arrival time and its scheduled playout time (no timestamp or scheduled playout time exist). Delay control actions should depend on the current buffer occupancy. A simple way to avoid large end-to-end delays is to play all MUs as soon as they arrive and forbid buffering. This is equivalent to using a *buffer-less* playout scheduler.

B. Buffer occupancy control: Queue monitoring and Watermark-based schedulers.

The work of Stone and Jeffay [29] demonstrates the fundamental idea that instead of measuring delay differences, one can directly measure the impact of the delay jitter on a receiver by observing the occupancy of the playout buffer over time. The proposed policy is called *queue monitoring* (QM). Under QM, a sequence of video frames that has been presented with no gap between frames – meaning that the queue was never found empty following the completion of a presentation – is used as an indication of reduced delay variability and triggers a reduction of the end-to-end delay of the stream by discarding the newest frame from the buffer. The selection of the duration for the gap-free interval regulates how aggressively QM tends to reduce latency and is thus the synchronization/latency tradeoff parameter. The implemented system uses a series of thresholds (for every occupancy level) and associated counters for the derivation of the frame discard decisions. Increasing network jitter causes buffer underflows (display gaps) and naturally increases the occupancy – thus the buffering delay – with the acceptance and presentation of "late" frames. That is, QM is to some extent *data-preserving*, as in some occasions presents late frames.

QM uses a window mechanism for the adjustment of delay, however, many buffer oriented schedulers are given the freedom to adjust the playout point in a per-MU fashion [30], [31], [5], [32], [33], [6]. Rothermel and Helbig [30] introduce the idea of occupancy *watermarks* (high-watermark, HWM and low-watermark, LWM) to define a range of desired playout buffer occupancies that balance the risk between buffer underflow and buffer overflow (see Fig. 5). A *targeted area*, lying between the HWM and the LWM levels, is defined by the upper target boundary (UTB), and lower target boundary (LTB). The positioning and the width of the targeted area (inside the watermark limits) reflects the desired synchronization/delay compromise. For example, a minimum-delay policy sets the LTB equal to the LWM and the UTB to a slightly larger value. When the occupancy of the buffer falls outside the targeted area – in the so-called *critical buffer regions* – the scheduler enters an adaptation phase with the aim of returning the occupancy inside the targeted area. This is accomplished by modifying the receiver's consumption rate

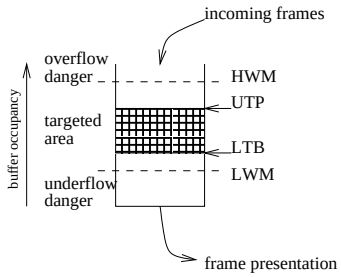

Fig. 5. The watermark-based playout scheduler of Rothermel and Helbig. The selection of watermarks mainly affects the underflow and overflow probabilities. The positioning of the target area, inside HWM and LWM, regulates the tradeoff between intrastream synchronization and stream latency.

until the occupancy returns in the targeted area. The width of the targeted area determines the aggressiveness of the buffer control algorithm, while the selection of watermarks mostly contributes to the data loss rate. The watermarks are fixed in [30] but it is noted that the scheduler can be made adaptive by dynamically regulating the watermarks in response to jitter effects (e.g. MU-loss).

Biersack et al. [31] have modified the watermark method of Rothermel [30], by placing the rate adaptation mechanism at the sender (which is a VOD server) rather than at the receiver. The receiver uses a gradient descent estimator of the buffer occupancy in order to smooth out occupancy fluctuations caused by short term jitter. When the smoothed buffer occupancy falls in the *critical region* (outside LWM,HWM), the receiver sends a signal to the sender suggesting that the latter should adjust its rate so that the smooth occupancy returns in the targeted area. The sender either skips some frames or pauses (pause/drop method) to adjust its rate. Source rate adaptation that affects the encoding process[6] could also be used, but the authors do not consider it due to high implementation complexity.

C. Buffer occupancy control with dynamic regulation of the duration of MUs.

Most playout schedulers (time/buffer oriented) have followed a *slotted* approach in the regulation of the buffering delay, in the sense that they increase or decrease it in constant amounts (slots) that equal the duration of a MU. The discard of "late" frames [31], [34] and the *tail-drop* from over-built queues [29] lead to sharp delay reduction jumps of duration T, equal to the duration of a video frame. Similarly, when the playout buffer empties, the presentation resumes after one or more MU periods [31]. This approach owes its popularity to implementation simplicity but can be quite crude, especially in the case of low frame rate streams where the slot (video frame) has a significant duration. Take for example a low bit rate 15 frames/sec encoder with each frame lasting 66 msec. Pause/drop operations on that slot scale can significantly degrade the perceptual quality, as it is easier for the human visual system to detect disruptions of such coarse magnitude.

The work of Yuang et al. [5], [32] has demonstrated an improvement in the perceptual quality of video, achieved by a fine-grained regulation of playout durations based on the current occupancy of the playout buffer. In [5] a *threshold-based* playout scheduler is proposed; the scheduler employs progressively reduced playout rates, aiming at avoiding large underflow discontinuities, as the buffer occupancy i drops below a threshold value TH. The selection of TH is made prior to stream initiation and remains unchanged despite jitter fluctuations; it governs the tradeoff between stream continuity and reduction of playout rate. Stream continuity is described by two disjoint metrics: the probability of an empty buffer, and the frame loss probability due to buffer overflow. The work has been enhanced in [32] by introducing a dynamic playout scheduler which uses a *window* to optimize some quality metric by responding to changing network delay jitter conditions. The window is in essence a time-varying, dynamic version of the threshold approach. A neural network (NN) traffic predictor and a NN window determinator are being used for the online estimation of traffic characteristics and the regulation of the window size. The derived value for the window is compared to the current buffer occupancy, resulting in the selection of playout durations for the buffered frames. Stream continuity is described by a second-order metric – the variance of discontinuity (VoD) – which accounts for underflow occurrences and discontinuities due to reduced playout rates. A number of playout schedules are derived, each providing a different tradeoff between playout continuity (captured by VoD), and reduction of mean playout rate.

The threshold-based scheduler of [5] has been extended in Laoutaris and Stavrakakis [33]. A compact and fair continuity metric has been introduced. It is called the *Distortion of Playout* (DoP) and combines all causes of media asynchrony: underflow gaps, slowdown gaps, and overflow gaps – in a concise, and experimentally justified way. The definition of DoP has been motivated by experimental perceptual results for video transportation over packet networks – conducted by Claypool and Tanner [9] – reporting that jitter degrades the perceptual quality of video nearly as much as packet loss does. The study has limited the range of the threshold parameter, TH, by identifying a range of values where there is no beneficial tradeoff between continuity and reduction of mean playout rate – the two antagonistic metrics of interest. Interestingly, it has been shown that this range of values changes with the burstiness of the frame arrival process[7], revealing the danger of an initially meaningful TH appearing in the undesirable area due to a change of the arrival burstiness. Finally, the work is supplemented with online algorithms for the detection and the maintenance of the operational parameter TH within the area of beneficial tradeoff across unknown, non-stationary, delay jitter.

The work of Laoutaris and Stavrakakis [6] develops a scheduler that outperforms the earlier empirical schedulers of [5], [33] by solving an appropriate optimization problem. Stream continuity is described by using the first two moments of the Distortion of Playout metric. This approach allows for a fine grained optimization of stream continuity by catering to a combination of both the expected frequency of synchronization loss and its appearance pattern. It is realized that the minimization of the expected value of DoP and the minimization of the variability

[6] Dynamically adjust the number of quantization levels in order to reduce the output bit rate of the encoder.

[7] Burstiness is created by delay jitter that transforms a periodic stream to a bursty stream consisting of clustered arrivals and large inactivity periods.

author	buffer control	delay performance	intrastream sync. metric	evaluation
Stone and Jeffay (QM) [29]	tail dropping	try to reduce	gap freq., 1st order	experimental
Rothermel and Helbig [30]	adjust playout rate	predefined(static tradeoff)	gap prob.,1st order	simulation
Biersack et al. [31]	adjust source rate	predefined(static tradeoff)	gap prob.,1st order	experimental
Yuang et al. [5]	static threshold	threshold dependent	gap prob.,1st order	analytical
Yuang et al. [32]	NN dynamic threshold	variable delay	2nd order	simulation
Laoutaris and Stavrakakis [33]	adaptive threshold	small initial,increase	2nd order	analytical
Laoutaris et al. [6], [35]	offline optimal policy	policy dependent	combined 1st,2nd order	analytical

TABLE II

OVERVIEW OF SURVEYED BUFFER-ORIENTED SCHEDULERS. IN THE BUFFER CONTROL COLUMN, A *slow* (*fast*) IN PARENTHESES, DENOTES THAT THE SCHEDULER IS ABLE TO APPLY REDUCED (INCREASED) PLAYOUT RATES.

of DoP, are two contradicting objectives. It is concluded that, for a perceptually optimal result, the scheduler must be allowed to increase the frequency of discontinuities, if this increase is to provide for a smooth spacing between discontinuity occurrences, and thus help in concealing them. Markov Decision theory is employed for the derivation of the optimal playout policy for a certain level of network jitter. This work has been extended in [35] to encompass different levels of network jitter. By calculating in advance (offline) the optimal playout policy for some common levels of network jitter, a playout scheduler can use a jitter estimator and adaptively "load" the appropriate, offline-computed, optimal policy and, thus, approach the optimal performance in a dynamic environment, at a low complexity (no online optimization required).

VI. DISCUSSION AND IMPLEMENTED SYSTEMS

Various concepts that were discussed during the presentation of individual systems are summarized in this section with the aim to address the appropriateness of each scheme for different types of applications. Some examples from implemented systems are drawn to support the discussion. The appropriateness of schemes for different media types was discussed in Sect. III. Here we discuss the delay performance and the complexity of the proposed schemes. In general, time-oriented schedulers are preferred when there is an interactivity requirement, in most cases in systems that handle spoken voice. Buffer-oriented systems are employed in video communication systems, where some compromise in delay is acceptable, even in interactive systems, if this is to provide for a smooth presentation of frames.

Bidirectional, interactive audio applications usually implement time-oriented playout schedulers. When interactivity is the major concern, a global clock is implemented and MUs are delivered at a constant low delay or are discarded. Many systems cannot afford the added complexity of a global timing mechanism (NTP or GPS) and choose to use differential delay methods. By doing so, they sacrifice the guarantee (in absolute values) for the total end-to-end delay. The RAT audio tool [36] implements the algorithms of Ramjee et al. [22] and Moon et al. [25] which have been described in Section IV-C (RAT uses $\beta = 3$ in equation 1).

Bidirectional video applications are usually less demanding, as far as interactivity is concerned, compared to their audio counterparts. The buffer-less approach is very simple and provides for the best interactivity (frames are displayed as soon as they arrive), but the synchronization quality quickly degrades with jitter, as there is no dejitter buffer. The VIC video conferencing tool [37] does not implement a dejitter buffer and transfers the video frames to the decoder upon arrival. The self-adjusting buffer is also quite simple to implement, and assuming a small amount of jitter, provides a very good synchronization/delay tradeoff as the buffer quickly adjusts to an occupancy that eliminates all jitter, at a small delay. The downside is that it is sensitive to rare occurrences of unusually large jitter; in such cases the delay of the stream will rise and will remain large since there is no delay control function to restore it. Neither scheme employs smoothing of long-lasting discontinuities. Dynamic regulation of MU durations can be used to improve the intrastream synchronization quality. The threshold-based schemes [5], [33] are quite simple to implement while systems that require some offline optimization are more complex [32], [6].

Streaming of stored content can be carried out by using the algorithms of Section IV-A which provide for an absolute resynchronization at the lowest possible end-to-end delay. Implemented systems seldomly use them though, mainly because these algorithms induce an initial delay that is up to the maximum network jitter (J_{max}) which in most cases is unknown. But even with a known J_{max}, different sides could argue against the assured synchronization. Some would consider an initial buffering delay that smoothes out J_{max} unnecessary, since J_{max} is seldomly experienced. On the other hand, when J_{max} is fairly small, a system can choose to prebuffer even more data than needed to smooth it out as this allows for the incorporation of selective retransmission protocols at the receiver, shielding the stream against network losses. The Real Player from RealNetworks Corp. implements a buffering scheme that favors the protection of stream continuity; a reasonable choice since the player is used for the reproduction of stored content so latency is less important. The player initiates the presentation of frames after it has downloaded the entire clip, or a significant part of it, up to the available free memory. In the latter case it is possible to run into buffer underflows, but these would occur only under extreme network jitter. The Media Player from Microsoft Corp. allows the user to set the amount of prebuffered data. A default value of 3-5 seconds is recommended; larger values protect the stream from even larger jitter, while smaller values make the service more interactive. A similar capability is also available as a choice in the Real Player, with a large suggested initial delay.

VII. RELATED RESEARCH ISSUES

This section briefly comments on two research areas that are of particular interest to playout adaptation, namely, forward er-

ror correction (FEC) and video caching (or proxying). FEC and its coupling with playout adaptation is a research topic that has recently attracted much attention, owing perhaps to the fact that FEC plays a significant role in enabling packet video communications in wireless environments. Video caching appears as a very attractive way to provide high quality, non-interactive streaming content in a cost-effective manner.

A. FEC and playout adaptation

Most playout adaptation algorithms consider packet losses due to packet lateness only, and disregard packet losses that occur in the network due to congestion. Network losses are assumed to be out of the scope of playout adaptation and the compensation of losses is left to various forward error correction mechanisms [38], [39], [40], which operate in isolation from the playout adaptation algorithm. Recent work has shown that a considerable performance gain can be expected from the coupling of the delay-oriented playout adaptation, and the loss-oriented FEC [41], [42].

Rosenberg et al. [41] study the effect of $(n - k)$ Reed-Solomon correction codes[8] on existing and new playout algorithms and show that performance improvement is achieved when considering the coupling between jitter and loss compensation. Existing playout algorithms [22], [25] are made FEC-aware by substituting the network delay (D_n) of a packet with the *virtual network delay* (VD_n) – which is either D_n, or the extended *recovery delay* (recovery time minus generation time). If no error occurs in the network, then the recovery time of a MU coincides with its arrival time, otherwise the recovery time is the time when the reception of some redundant FEC packet allows the correct decoding of the corrupted[9] (or lost) MU. The targeted end-to-end delay D_{tot} is shaped by VD_ns, which in turn depend on the FEC algorithm, thus the coupling. Several new adaptation algorithms have been proposed. The – *adaptively virtual* algorithm – targets a desirable packet loss probability. It is based on [22], and uses virtual delays to dynamically adjust the variation multiplier β of (1) in order to achieve a targeted loss rate. Another algorithm is the so called *Previous Optimal* algorithm; it determines the optimal (minimal) delay for the previous talkspurt, such that a specific application loss rate be maintained, and applies it to the next talkspurt, hence the *Previous* in the algorithm name. Finally, the authors describe an analytical framework for the expression of the total end-to-end delay as a function of the application level reception probability (in the presence of FEC), the network delay distribution, and the network loss probability.

Hartanto and Tionardi [42] also consider the interaction between media synchronization and error control. Given a fixed amount of bandwidth for a video stream, additional FEC packets protect the stream against network losses, but they also increase the transmission delay of packets. The increased overall delay may lead to loss due to packet lateness, so the end users may experience worse quality than in the case without FEC. The authors introduce the concept of *cumulative jitter* – which is the running total of time variations between the application-data-unit spacings at the receiver and at the sender – and use it to study the tradeoff between network loss and loss due to packet lateness. Forward error correction has also been studied in conjunction to source rate control [43].

Intrastream synchronization has been studied for wireless receivers's by Liu and Zarki [28]. In the wireless environment, the media synchronization module must be coupled with the ARQ of the wireless link.

B. Video caching and playout adaptation

As with traditional data objects of established protocols like http and ftp, real-time objects – mostly stored video – are being cached, or proxied, with the aim of reducing the network traffic and improving the interactivity of content delivery [44], [45], [46], [47], [48], [49] . Unlike the popular web proxies, video proxies typically store only a portion of a video clip, the initial part usually – called the *prefix* [44], [45], [46] – as the entire video object is too lengthy to be replicated on a typical proxy nor is replication the most cost-effective alternative [45]. Video proxies improve the quality of video delivery in many ways. First, they reduce the network transfer delay, as proxies are located closer to end clients, thus the stream travels just a few network hops before the delivery. Second, proxies assist in the improvement of intrastream synchronization quality. If the prefix of a video is of large duration, in the order of minutes, then the amount of proxied data completely smoothes out the jitter in the data path from the server to the proxy. This is important as it relieves the receiver from a large fraction of jitter; only the jitter in the access part remains, from the proxy to the receiver, which is usually small and can be easily smoothed out with a small playout buffer at the receiver. The reduction of the playout buffer reduces also the total delay, thus improving the responsivity of the service. Even if the amount of prefix corresponds to a time duration of the same time scale with the network jitter at the core network, it will again absorb some portion of the delay variability and, thus, decrease the size of the playout buffer at the receiver. For a given synchronization quality, the existence of the prefix, helps by "hiding" some portion of the total delay.

VIII. Conclusions

This paper has presented a survey of the literature on playout schedulers. Playout schedulers play a significant role in stream communications over best-effort networks as they protect the intrastream synchronization quality of the real-time stream. A significant amount of work has been published in this field, differentiating in a number of key aspects such as: the granularity of timing information, the handling of late media units, the employed metrics, and the adaptation to changing delay conditions. This paper has tried to provide a structured overview of these issues and taxonomize the proposed playout schemes accordingly.

[8] $(n - k)$ redundant FEC packets protect the k data packets. Any k of the n packets must be received to recover the remaining $n - k$ packets.

[9] Video applications usually fragment a frame as it is too large to fit in a single network packet; the loss of a single fragment corrupts the entire frame and calls for the use of redundant FEC info for its salvation.

References

[1] Julio Escobar Craig Partridge and Debra Deutsch, "Flow synchronization protocol," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, Apr.

1994.

[2] Changdong Liu, Yong Xie, Myung J. Lee, and Terek N. Saadawi, "Multipoint multimedia teleconference system with adaptive synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, Sept. 1996.

[3] John F. Gibbon and Thomas D. C. Little, "The use of network delay estimation for multimedia data retrieval," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, Sept. 1996.

[4] Louise Lamont, Lian Li, Renaud Brimont, and Nicolas D. Georganas, "Synchronization of multimedia data for a multimedia news-on-demand application," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, Jan. 1996.

[5] Maria C. Yuang, Shih T. Liang, Yu G. Chen, and Chi L. Shen, "Dynamic video playout smoothing method for multimedia applications," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Dallas, Texas, June 1996, p. S44.

[6] Nikolaos Laoutaris and Ioannis Stavrakakis, "An analytical design of optimal playout schedulers for packet video receivers," *submitted for publication in Computer Communications journal. An earlier version was presented at the 2nd International Workshop on Quality of Future Internet Services (QofIS2001), Coimbra, Portugal, 2001*, 2002.

[7] Ralf Steinmetz, "Human perception of jitter and media synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, Jan. 1996.

[8] G. Ghinea and J.P. Thomas, "Qos impact on user perception and understanding of multimedia video clips," in *ACM Multimedia '98*, Bristol, UK, 1998.

[9] Mark Claypool and Jonathan Tanner, "The effects of jitter on the perceptual quality of video," in *ACM Multimedia '99*, Orlando, FL, USA, 1999.

[10] Warren A. Montgomery, "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, pp. 1022–1028, Dec. 1983.

[11] Giulio Barberis and Daniele Pazzaglia, "Analysis and optimal design of a packet-voice receiver," *IEEE Transactions on Communications*, vol. COM-28, no. 2, pp. 217–227, Feb. 1980.

[12] William E. Naylor and Leonard Kleinrock, "Stream traffic communication in packet switched networks: destination buffering constraints," *IEEE Transactions on Communications*, vol. COM-30, no. 12, pp. 2527–2534, Dec. 1982.

[13] Harry Santoso, Dairaine Laurent, Serge Fdida, and Eric Horlait, "Preserving temporal signature: a way to convey time constrained flows," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, Houston, USA, Nov. 1993.

[14] Narayanan Shivakumar, Cormac J. Sreenan, B. Narendran, and Prathima Agrawal, "The concord algorithm for synchronization of networked multimedia streams," in *International Conference on Multimedia Computing and Systems*, 1995.

[15] D. Frankowski and J. Riedl, "Hiding jitter in an audio stream," Tech. Rep. TR-93-50, University of Minnesota Department of Computer Science, 1993.

[16] W. Geyer, C. Bernhardt, and E. Biersack, "A synchronization scheme for stored multimedia streams," *Lecture Notes in Computer Science*, vol. 1045, 1996.

[17] Mario Baldi and Yoram Ofek, "End-to-end delay analysis of videoconferencing over packet switched networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.

[18] Vern Paxson, "End-to-end internet packet dynamics," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Cannes, France, Sept. 1997.

[19] Sunil Kalidindi and Matthew J. Zekauskas, "Surveyor: An infrastructure for internet performance measurements," in *Proc. of INET*, San Jose, California, June 1999.

[20] David L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[21] C.J. Sreenan, Jyh-Cheng Chen, P Agrawal, and B Narendran, "Delay reduction techniques for playout buffering," *IEEE Transactions on Multimedia*, vol. 2, no. 2, June 2000.

[22] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994, pp. 680–688, IEEE Computer Society Press, Los Alamitos, California.

[23] Van Jacobson and Michael J. Karels, "Congestion avoidance and control," in *SIGCOMM Symposium on Communications Architectures and Protocols*. ACM, Nov. 1998.

[24] Jean-Chrysostome Bolot, "End-to-end packet delay and loss behavior in the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Deepinder P. Sidhu, Ed., San Francisco, California, Sept. 1993, ACM, pp. 289–298, also in *Computer Communication Review* 23 (4), Oct. 1992.

[25] Sue B. Moon, Jim Kurose, and Don Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *ACM/Springer Multimedia Systems*, vol. 5, no. 1, pp. 17–28, Jan. 1998.

[26] Marco Roccetti, Vittorio Ghini, Giovanni Pau, Paola Salomoni, and Maria Elena Bonfigli, "Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the internet," *Multimedia Tools and Applications*, vol. 14, no. 1, May 2001.

[27] Felipe Alvarez-Cuevas, Miquel Bertran, Francesc Oller, and Josep M. Selga, "Voice synchronization in packet switching networks," *IEEE Network*, vol. 7, no. 5, pp. 20–25, Sept. 1993.

[28] Hang Liu and Magda El Zarki, "Delay and synchronization control middleware to support real-time multimedia services over wireless PCS networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, pp. 1660–1671, Sept. 1999.

[29] Donald L. Stone and Kevin Jeffay, "An empirical study of a jitter management scheme for video teleconferencing," *Multimedia Systems*, vol. 2, no. 2, 1995.

[30] Kurt Rothermel and Tobias Helbig, "An adaptive stream synchronization protocol," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Durham, New Hampshire, Apr. 1995, Lecture Notes in Computer Science, pp. 189–202, Springer.

[31] Ernst Biersack, Werner Geyer, and Christoph Bernhardt, "Intra and inter-stream synchronisation for stored multimedia streams," in *ICMCS (IEEE Multimedia Conference)*, Hiroshima, Japan, June 1996.

[32] Maria C. Yuang, Po L. Tien, and Shih T. Liang, "Intelligent video smoother for multimedia communications," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 136–146, Feb. 1997.

[33] Nikolaos Laoutaris and Ioannis Stavrakakis, "Adaptive playout strategies for packet video receivers with finite buffer capacity," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Helsinki, Finland, June 2001.

[34] Pawan Goyal, Harrick M. Vin, Chia Shen, and Prashant J. Shenoy, "A reliable, adaptive network protocol for video transport," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Fransisco, California, Mar. 1996.

[35] Nikolaos Laoutaris, Benny Van Houdt, and Ioannis Stavrakakis, "Optimization of a packet video receiver under different levels of delay jitter: An analytical approach," *submitted work*, 2002.

[36] Isidor Kouvelas and Vicky Hardman, "Overcoming workstation scheduling problems in a real-time audio tool," in *Proc. of Usenix Winter Conference*, Anaheim, California, Jan. 1997.

[37] Steve McCanne and Van Jacobson, "vic: A flexible framework for packet video," in *Proc. of ACM Multimedia '95*, Nov. 1995.

[38] Georg Carle and Ernst W. Biersack, "Survey of error recovery techniques for IP-based audio-visual multicast applications," *IEEE Network*, vol. 11, no. 6, pp. 24–36, Nov. 1997.

[39] Colin Perkins, Orion Hodson, and Vicky Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, no. 5, pp. 40–48, Sept. 1998.

[40] Hang Liu, Hairuo Ma, Magda El Zarki, and Sanjay Gupta, "Error control schemes for networks: An overview," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 167–182, Oct. 1997.

[41] Jonathan Rosenberg, Lili Qiu, and Henning Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Tel Aviv, Israel, Mar. 2000.

[42] Felix Hartanto and Laurensius Tionardi, "Effects of interaction between error control and media synchronization on application-level performances," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, 2000, pp. 298–303.

[43] Jean-Chrysostome Bolot, Sacha Fosse-Parisis, and Don Towsley, "Adaptive FEC-Based error control for interactive audio in the Internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[44] Subhabrata Sen, Jennifer Rexford, and Don Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[45] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley, "Proxy-based distribution of streaming video over unicast/multicast connections," Technical Report UMASS TR-2001-05, University of Massachusetts, Amherst, 2001.

[46] S.-H. Gary Chan and Fouad A. Tobagi, "Caching schemes for distributed video services," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Vancouver, Canada, June 1999.

[47] J. Chuang, "Distributed network storage service with quality-of-service guarantees," *Journal of Network and Computer Applications*, 2000.
[48] Jussi Kangasharju, Felix Hartanto, Martin Reisslein, and Keith W. Ross, "Distributing layered encoded video through caches," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Anchorage, Alaska, Apr. 2001.
[49] Zhi-Li Zhang, Yuewei Wang, D. H. C. Du, and Dongli Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.