# Optimization of a Packet Video Receiver Under Different Levels of Delay Jitter: An Analytical Approach*

Nikolaos Laoutaris[†], Benny Van Houdt[‡], and Ioannis Stavrakakis[§]

**Abstract**

This paper studies the problem of analyzing and designing optimal playout adaptation policies for packet video receivers (PVR) that operate in a delay jitter inducing best-effort network, like the current internet. The developed system model is built around the $E_k/D_i/1/N$ phase-type queue and allows for the effective modeling of key design and system parameters, such as: the level of delay jitter, the performance metrics and the employed playout policy. The optimal playout policy is derived under k-Erlang interarrivals by formulating and solving an optimization problem. The (theoretical) optimal solution is transformed into an approximately optimal one that utilizes observable information and it is, thus, feasible. Numerical results are derived under the optimal policy and compared against those under the optimal policy that assumes a fixed level of jitter as determined by Poisson arrivals, as well as against the deterministic service that applies no playout adaptation. Based on this work, a PVR is proposed that adapts to varying network delay jitter and tries to induce a performance that approximates the derived theoretical optimal one.

## 1 Introduction

The transmission of video streams, real-time or pre-stored, 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 the periodically transmitted video frames thus hinders the comprehension of the stream. Playout adaptation algorithms undertake the labor of the temporal reconstruction of the stream to a form that resembles as much as possible its initial structure, as enforced by the encoding process. The *intrastream synchronization* quality of the stream quantifies the extent of

*Work supported in part by the Special Account for Research Grants of the National and Kapodistrian University of Athens and the IST program of the European Union under contract IST-2001-32686 (Broadway).

[†]Nikolaos Laoutaris is with the Dept. of Informatics and Telecommunications, University of Athens, 15784 Athens, Greece. E-mail: laoutaris@di.uoa.gr

[‡]Benny Van Houdt is a postdoctoral Fellow of the FWO Flanders and is with the Dept. of Math. and Computer Science, University of Antwerp, B-2020 Antwerp, Belgium. E-mail: benny.vanhoudt@ua.ac.be

[§]Ioannis Stavrakakis is with the Dept. of Informatics and Telecommunications, University of Athens, 15784 Athens, Greece. E-mail: istavrak@di.uoa.gr

this temporal reconstruction and is directly related to the perceived presentation quality at the receiving end.

A packet video receiver (PVR) consists of a playout buffer, for the temporary storage of incoming frames, and a playout scheduler, for the determination of the presentation initiation time and the presentation duration of each frame. The playout scheduler is given the ability to regulate the presentation duration of a video frame (which normally is fixed and equal to the inverse of the frame production rate) in an attempt to smooth-out the effects of network jitter. The general principle that drives the operation of the scheduler is that large discontinuities between consecutive frames are undesirable as they are easily detected by human users and, therefore, it is desirable to break them into discontinuities of smaller duration that may be unnoticed due to human perceptual limitations in the detection of motion.

By manipulating the duration of frames, the playout scheduler also affects the number of buffered frames. Unpresented frames that wait in the playout buffer increase the end-to-end delay of each newly arriving frame. The end-to-end delay measures the time between the encoding of a frame at the sender and its presentation at the receiver. Applications that have a dialogic nature (e.g., videoconferencing) call for a small end-to-end delay so that they can offer the required interactivity to the communicating parties.

This paper is concerned with the performance evaluation and optimization of *buffer-oriented* playout schedulers, which perform their task without the use of timing information (clocks and timestamping of frames), as opposed to *time-oriented* schedulers which utilize such information. The systems that are considered here use the current occupancy of the playout buffer as an implicit indication of jitter and base all regulatory actions on that information. The two main contributions of this work are: the development of an analytical performance evaluation model for packet video receivers, capturing key design and environmental parameters such as the level of delay jitter, the operation of the playout scheduler, and the considered quality metrics; and the development of an analytical optimization model that has the potential of deriving the optimal PVR design, under appropriate quality metrics, for different levels of delay jitter.

The remainder of the paper is organized as follows. In Sect. 2 we examine some relevant work from the literature. In Sect. 3 we discuss issues related to the modeling of frame arrivals at the PVR. A queuing model for PVRs and the associated performance metrics are developed in Sect. 4. Section 5 formulates a Markov Decision problem whose solution is the theoretical optimal PVR for a given level of delay jitter. Some numerical results from the optimized systems along with a comparison with earlier systems are presented in Sect. 6. In Sect. 7 we show how to apply the theoretical optimal solution to a real world PVR. In Sect. 8 we describe the overall architecture of a potential implementation of the system and propose a way to adapt to fluctuating delay jitter.

## 2 Related work

A survey of proposed playout schedulers, both time and buffer oriented, has been presented in [1]. Here we selectively present some buffer-oriented schemes that are of particular interest to the current work.

The fundamental idea that the level of delay jitter can be implicitly deduced by observing the occupancy of the playout buffer has been demonstrated with a system that implements the *queue monitoring* (QM) algorithm [2]. Under QM, a sequence of video frames that has been presented in a continuous manner – 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 configuration of the algorithm is empirical, based on traces of real frame interarrivals. Although QM handles the basic continuity-latency tradeoff it does not try to "smooth out" the disruptive effects of this process. It allows the natural build-up of the buffer with (detectable[1]) underflows, while it decreases the delay with frame discards (which can also be detectable, especially in the case where the frame has a significant duration, e.g., in low frame rate encoding).

Newer systems try to avoid long lasting discontinuities . The *threshold-slowdown* scheduler of [3] applies a more general regulation scheme governed by the selection of the slowdown-threshold $TH$. Frames are presented with a normal duration when the buffer occupancy, $i$, is greater than the (fixed) threshold, and with extended duration, by a factor equal to $TH/i$, when the buffer occupancy is smaller than the threshold. In essence, the scheduler attempts to prevent an impending underflow, when the occupancy of the buffer is small, by applying gradually reduced playout rates. In [4] we examined some of the implications of network jitter in the selection of an appropriate threshold value and provided algorithms that modify this value on the fly, in response to changing jitter.

The work that is most relevant to the current is [5]. It differs from [3, 4] in that instead of using a heuristic method (like the aforementioned threshold) for the regulation of the playout rate, it applies the best possible playout policy (for the assumed environment) which emanates from the analytical solution of an appropriate optimization problem. This playout policy, however, delivers the desired optimal performance only under the assumed level of delay jitter. One contribution of the present work is that it extends the methods and the results of [5] so that they may be applied to different levels of delay jitter, allowing for the exploitation of the system in real-world PVRs that operate under fluctuating delay jitter.

# 3 Modeling delay jitter

Video frames are periodically transmited by a sender at a rate $\lambda_f$ which is specific to the employed video format, usually at 25 or 30 frames/sec. The spacing between consecutive frames at the sender and the duration of each frame, $T$, are equal, given by the inverse of the frame production rate, i.e, $T = 1/\lambda_f$.

Due to the variable network transfer delay of best-effort networks, frames arrive at the PVR at non-regular intervals that may deviate significantly from the frame period $T$. The variability of the interarrival intervals is directly related to the variability in the network transfer delays (network jitter). The $i$th frame interarrival, $X_i$, is given by: $X_i = T + D_{n,i} - D_{n,i-1}$, where $D_{n,i}$ denotes the network delay of the $i$th frame. If $D_{n,i} = D_{n,i-1}$ the interarrival spacing is equal to the interdeparture spacing. If $D_{n,i} > D_{n,i-1}$ the two frames drift apart ($X_i > T$) otherwise they approach each other ($X_i < T$). For $D_{n,i-1} = D_{n,i} + T$ the two frames arrive concurrently at

[1]We mean "detectable" under the motion detection capabilities of a human end-user.

the PVR, a phenomenon called clustering of frames. Due to the high degree of aggregation of the traffic that co-exists with the video stream in the network, it becomes reasonable to assume independent and identically distributed (i.i.d.) network transfer delays, that is, $D_{n,i}$s, thus giving rise to the following observations: the expected duration of interarrivals is $E\{X\} = T$; the variance of interarrivals is $Var\{X\} = 2 \cdot Var\{D_n\}$; the distribution is symmetrical around its mean value. See [6] for more details on these observations.

Previous studies of buffer-oriented playout schedulers have used the Poisson [3, 4, 5, 7], or the interupted Poisson process [8, 4], for the modeling of frame arrivals. The exponentially (hyperexponentially) distributed interarrivals, that are implied by the Poisson (interrupted Poisson) process, have some properties that limit their value as models of the true interarrivals of periodic streams that have been reshaped by jitter. First, the exponential distribution is not symmetrical around its mean value, as required by the independence of $D_{n,i}$s. The symmetrical nature of the interarrival times does not only stem from the i.i.d. assumption made for the network delays but it has also been verified experimentally on real networks [6, 9, 10]. Second, the exponential distribution is much more variable than measured interarrival distributions, making it appropriate only under conditions of extreme delay jitter that results in highly variable interarrivals at the PVR. These observations apply, to a greater extent, also for the interrupted Poisson process.

Under "normal" network conditions, frame interarrivals tend to be much more regular than what the exponential distribution provides. To capture this increased regularity we use throughout the rest of this work the $k$-Erlang distribution for the modeling of frame interarrivals. A $k$-Erlang distribution, being a $k$-fold convolution of an exponential distribution, is $k$ times *more regular* than the exponential distribution of the same mean. A $k$-Erlang interarrival $X$ with mean value $T$ is given by: $X = \sum_{i=1}^{k} Y_i$, where $Y_i$, for $1 \leq i \leq k$, is an exponentially distributed random variable with mean $T/k$. For the random variables $X$ and $Y$ we have: $E\{X\} = k \cdot E\{Y\}$=T; $Var\{X\} = k \cdot Var\{Y\} = T^2/k$; $Var\{Y\}/E^2\{Y\} = 1/k$. The last ratio denotes the regularity of the distribution. The exponential distribution has a reference regularity equal to one and is $k$ times less regular than the corresponding $k$-Erlang of same mean. The $k$-Erlang can approach the regularity of a deterministic distribution by increasing $k$ so that the ratio approaches 0. Also for sufficiently large $k$ the $k$-Erlang is almost symmetrically distributed around its mean value.

To identify the range of interarrival variability that should be modeled by the $k$-Erlang distribution we have transmitted a periodic stream of "dummy" frames, at 30 frames/sec, and have logged the interarrivals at the receiving host. The stream has crossed the data path from the University of Athens (UoA), Greece, to the Arizona State University (ASU), USA. The Mgen/Drec suite of tools [11] was used for the creation of the test traffic and for the logging of the interarrival trace. Figure 1 illustrates the measured variance of interarrivals at ASU, over 10 minute intervals, throughout an entire working day. In the y-axis instead of marking the actual measured variance, we mark the points that correspond to the variance of $k$-Erlang for $1 \leq k \leq 32$. The results indicate that the measured variance corresponds to a range of regularities from Poisson (high jitter at 10:00-11:00) down to 34-Erlang (much more regular interarrivals), with several intermediate levels in-between.

Although $k$ times more regular than Poisson, a $k$-Erlang input stream in the aforementioned range would lead to poor playout quality if not handled by an appropriate playout algorithm, such as the one developed here. In the following we provide an example to support this claim using

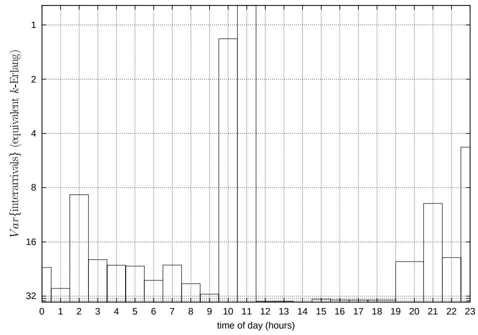


Figure 1: The variance of interarrivals of a test stream (with 30 frames/sec) from UoA to ASU. The x-axis marks the beginning time of each 10 min trace. The y-axis marks the points that correspond to the variance of interarrivals of $k$-Erlang distributed interarrivals (with 30 frames/sec mean rate). The logged interarrival correspond to jitter levels from slightly higher than Poisson, to as low as 34-Erlang.

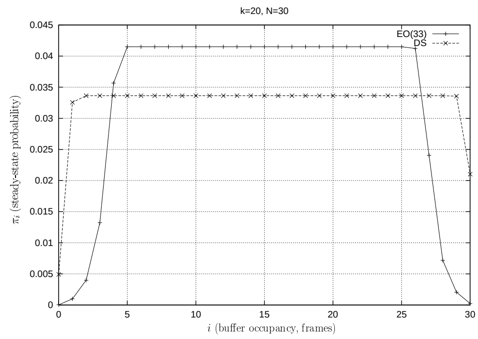


Figure 2: Steady-state buffer occupancy distribution for a receiver that can hold up to 30 frames. The input process is 20-Erlang. DS and EO(33) playout policies are considered.

$k = 20$. This value is a frequently encountered one in the measurement experiment. Figure 2 shows the steady-state buffer occupancy distribution of a receiver that is fed by a 20-Erlang input. The receiver can hold up to $N = 30$ frames. Two cases are ploted: (i) DS, which amounts to a deterministic service, where all frames are presented at their normal duration; (ii) EO(33), which is an example of the playout policies developed here and aims at avoiding discontinuities due to buffer underflows and overflows. The occupancy distribution is derived for frame presentation completion instances. Observe that under DS a 0.5% of presented frames are followed by an underflow, whereas the corresponding percentage for EO(33) is almost equal to zero. The 0.5% underflow percentage of DS amounts to $30 \cdot 60 \cdot 0.005 = 9$ underflows per minute for a 30 frames/sec stream. Such a rate of underflows is considered to be very high [2] thus resulting in poor playout quality. To this rate of discontinuities, one should add a substantial rate of buffer overflows. Indeed, observe that the DS playout often leads to high occupancies – where overflows occur – whereas the EO(33) policy effectively avoids high occupancies, thus avoiding frame overflows.

In the sequel we develop some new playout policies by using the $k$-Erlang distribution as a model which matches the first two moments of real frame interarrival distributions. It should be noted that the current work does not strive to model frame interarrivals as accuretly as possible, as has been done by traffic modeling works [12, 13]. The final objective here is to identify the optimal playout policy for different levels of delay jitter, thus, only the macrodynamics (first two moments) of the input traffic are modeled, therefore, Erlang arrivals suffice. It is possible to use a more elaborate interarrival process like a PH[14] or a (B)MAP[15, 16], though it remains to be seen whether the optimal playout policy can be determined by the value iteration algorithm in an efficient manner.

The numerical results of Sect. 6 are based on jitter levels that correspond to $k$-Erlang distributions, $1 \leq k \leq 50$, motivated by the measured variances in the UoA-ASU experiment. The selected range, however, need not be taken as a "standard" range of jitter fluctuation; other ranges of delay jitter, corresponding probably to even more regular arrivals ($k > 50$) could be examined as required by the targeted network environment. In fact we have obtained the optimal playout policies for much larger $k$ (above 150) that correspond to almost deterministic interarrivals. The scalability of the proposed algorithm for obtaining the optimal playout policy should not pose a problem as it is unnecessary to optimize for very large $k$, e.g., $k = 1000$, since for much smaller $k$ the optimal playout policy already reduces to deterministic playout (all frames are presented at their normal duration).

A final note on the jitter that is reported by our measurements is that this is only the network part of the overall jitter that is present at the application layer of a PVR. The end-system jitter, introduced by the operating systems of the end-systems, is missing. This jitter conponent in many cases can be quite large, possibly dominating the network portion of jitter, especially when overloaded video-on-demand servers are used for the streaming of the content. This means that even a small network jitter does not necessarily make the proposed playout policies obsolete, as there is still the end-system jitter that needs to be smoothed out.

# 4 Performance analysis of packet video receivers under $k$-Erlang arrivals

This section develops an analytical model that has the potential to capture the characteristics of a considerable number of buffer-oriented playout schedulers and provide for their performance evaluation across different levels of delay jitter. Along with the introduction of the queueing model, the involved performance metrics are presented and justified in a subsequent section. The analysis is carried out at the application layer of a PVR; it focuses on solid video frames that become available at the application layer from the underlying layers and uses the interarrival of frames at the application layer to capture the aggregate effect of the end to end delay jitter (network and end-system parts). Implementation specific issues such as video encoding and network level packetization schemes are not discussed in order to preserve the generality of the proposed playout policies. It is expected that with minor modifications the proposed algorithms should be applicable to a variety of encoding schemes (raw, MPEG, H263) and packetization formats. Finally, it is noted that although network packet losses are not considered in this work their effect on the overall stream quality is expected to be much smaller as compared to that of delay jitter. Loguinov and Radha confirm this claim in their recent large scale study of internet dynamics and its effect of video streaming [17] where it is reported that 98.9% of buffer underflow events in a PVR were due to delay jitter and only the tiny remainder due to packet loss[2]. Based on these reports and accounting that lost packets can be recovered/concealed by employing forward error correction techniques at the receiver it is believed that the presented jitter oriented assessment of playout quality will approximate sufficiently the quality in an operational system in the presence of some packet loss.

In the following a PVR is modeled as an $E_k/D_i/1/N$ queueing system, i.e., a queue with the following properties: an Erlang arrival process ($E_k$), appropriate for the modeling of jitter-dependent frame interarrivals; a deterministic state-dependent playout policy ($D_i$), modeling the operation of a general buffer-oriented playout scheduler which applies frame durations that depend on the current queue occupancy $i$; a finite playout buffer for $N$ video frames. In the next section, we obtain the steady state occupancy distribution for the $E_k/D_i/1/N$ queue upon service initiation times[3] by using the method of phases [18]. This method, commonly used to analyze the $E_k/D/1/N$ queue which is one of the simplest queues in the family of $PH/G/1$ queues [19], is straightforward to generalize to the $E_k/D_i/1/N$ queue.

## 4.1 The embedded Markov Chain

Frame interarrivals are assumed to follow a $k$-Erlang distribution with mean rate $\lambda_f = 1/T$, i.e., the $n$th interarrival $X_n$ is composed of $k$ i.i.d. intervals $Y_j$, all following the exponential distribution with parameter $\lambda = k\lambda_f$, such that: $X_n = \sum_{j=1}^{k} Y_j$. The passage of each exponential interval $Y_j$ is referred to as "completion of a phase" – with a single frame arrival occurring when $k$ phases have been completed.

Let $\{I_n\}_{n>0}$ denote the number of frames in the buffer at time $t_n$ which is the time prior to the presentation of frame $n$ and let $\{\tilde{I}_n\}_{n>0}$ be the number of phases in the system at $t_n$. $\{\tilde{I}_n\}_{n>0}$ counts the number of phases in the system accounting for the buffered frames (each frame is seen as a batch of $k$ phases) and the number of phases thus far completed by the ongoing arrival. The value of $I_n$ can be directly obtained from the value of $\tilde{I}_n$, which has $I_n$ embedded in it. Formally, $\tilde{I}_n = kI_n + J_n$, where $\{J_n\}_{n>0}$ is a discrete time stochastic process that counts the number of phases completed by the arrival process in the interval from $a_n$, the time of the last arrival prior to $t_n$, up to $t_n$ (see Fig. 3). The knowledge of $\tilde{I}_n$ not only provides for the exact number of frames in the buffer prior to the presentation of the $n$th frame (since $I_n = \lfloor \tilde{I}_n/k \rfloor$), but also adds information concerning the next arriving frame. The extra information (memory) is used for the approximation of non-memoryless interarrival distributions, such that are typical of periodic streams that have been reshaped by network jitter. Finally, it is important to notice that $\tilde{I}_n$ is a Markov chain, that is, $\tilde{I}_{n+1}$ does not depend upon $\tilde{I}_m$, for $m < n$, provided that $\tilde{I}_n$ is known.

Next, we indicate how to obtain the transition probability matrix $P$ of the Markov chain $\tilde{I}_n$. The value of $\tilde{I}_n$ varies between $k$ and $(N+1)k-1$: $k$ is the minimum number of phases (corresponding to one complete frame) that must be in place in buffer for the next presentation to begin; $(N+1)k-1$ phases correspond to the case of a full buffer ($N$ complete frames which are represented by $kN$ phases) and $k-1$ phases having been completed by the ongoing arrival. The evolution of the number of phases in the system can be seen as a queue with a waiting room for $(N+1)k-1$

[2]In their study they have used retransmissions for the recovery of lost packets but report that even without them the main disruptive cause would still be the delay jitter not the packet losses.

[3]Hereafter also called *presentation initiation* or *decision* instances, depending on the focus of the discussion.

customers – where each phase represents a customer – with the following characteristics: (1) the customers arrive according to a Poisson process with rate $k\lambda_f$; (2) the customers are served in batches of size $k$, where the service time $D_i$ of a batch is deterministic and depends on the number of phases $i$ found in the waiting room prior to service initiation; (3) if a customer finds the queue full upon arrival, it is dropped and $k-1$ other customers immediately leave the queue. Notice that the server serves the customers in batches of size $k$, therefore, it will not commence until there are at least $k$ customers in the waiting room. This requirement, along with the fact that the system is observed upon frame presentation initiation instants, mean that the system can never be found empty (with less than $k$ phases) upon an observation instant.

With the aforementioned system description, the expression of the phase transition probabilities, denoted as $p_{ij}(D_i) = Prob\{\tilde{I}_n = j | \tilde{I}_{n-1} = i, D_i\}$, is simplified. $D_i$ denotes the service duration for the $k$-sized batches; it depends solely on the number of phases in the waiting room prior to service initiation. Before proceeding, we present three examples of deterministic service disciplines: the deterministic service (DS) with constant duration $T$ independent of phase occupancy; the threshold slowdown (TS) algorithm of [3]; and the Poisson-optimal (PO) service discipline derived in [5] which selects a duration $\Delta_n$[4]: $1 \le n \le N$, $n$ being the frame occupancy (equal to $n = \lfloor i/k \rfloor$ here).

$$D_i = \begin{cases} T \quad ,k \le i \le (N+1)k-1 & \text{(DS)} \\ \max(\frac{TH}{\lfloor i/k \rfloor} \cdot T, T) \quad k \le i \le (N+1)k-1 & \text{(TS)} \\ \Delta_{\lfloor i/k \rfloor} \quad ,k \le i \le (N+1)k-1 & \text{(PO)} \end{cases} \tag{1}$$

Also note that the QM algorithm [2] (briefly discussed in Sect. 2) can be studied with the aforementioned model by appropriately augmenting the definition of the state so that in addition to the current occupancy it also includes the number of continuous uninterrupted frame playouts.

Based on the three characteristics of the size $(N+1)k-1$ queue mentioned above, we get the following transition matrix $P$, where the $(i,j)^{th}$ element of $P$ is denoted as $p_{ij}(D_i)$:

$$p_{ij}(D_i) = \begin{cases} \sum_{\sigma=0}^{2k-i} P\{\sigma, D_i\} & k \le i < 2k, j = k \\ P\{j-i+k, D_i\} & k \le i < 2k, k < j \le Nk-1 \\ \sum_{\sigma=0}^{\infty} P\{j-i+k+\sigma k, D_i\} & k \le i < 2k, Nk \le j \le (N+1)k-1 \\ P\{j-i+k, D_i\} & 2k \le i \le (N+1)k-1, i-k \le j \le Nk-1 \\ \sum_{\sigma=0}^{\infty} P\{j-i+k+\sigma k, D_i\} & 2k \le i \le (N+1)k-1, Nk \le j \le (N+1)k-1 \\ 0 & \text{elsewhere} \end{cases} \tag{2}$$

[4]The value of $\Delta_n$, for $1 \le n \le N$, is derived as the solution of an appropriate Markov decision problem assuming Poisson frame arrivals and studying the system upon service completions.

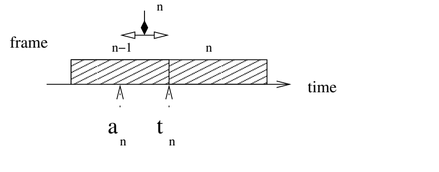


Figure 3: Timing diagram: $t_n$, the time prior to the presentation initiation of frame $n$; $a_n$, time of last arrival prior to $t_n$; $E_n$, the elapsed time since the last arrival, $E_n = t_n - a_n$.

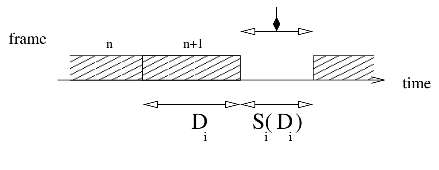


Figure 4: Time diagram of an underflow occurring during the presentation of frame $n+1$. As a consequence of the underflow, frame $n+1$ remains on display for a total duration equal to $D_i + S_i(D_i)$.

where $P\{m, X\}$ is the Poisson distributed probability of $m$ new phase arrivals occurring in an interval of duration $X$. The first five lines of equation (2) correspond to the following cases: (i) there is only one frame available in the buffer (at time $t_n$) and an underflow follows its presentation at $t_n + D_i$ (playout recommences when $k$ phases become again available thus $j = k$); (ii) there is only one frame available in the buffer (at time $t_n$) and no overflow(s) occur during its presentation; (iii) there is only one frame available in the buffer (at time $t_n$) and possible overflow(s) occur during its presentation; (iv) more than one frames are available in the buffer (at time $t_n$) and no overflow(s) occur during the imminent presentation; (v) more than one frames are available in the buffer (at time $t_n$) and possible overflow(s) occur during the imminent presentation.

Let $\tilde{\pi}_j$, $k \leq j \leq (N+1)k - 1$, be the steady-state probabilities of $\{\tilde{I}_n\}_{n>0}$; then $\pi_i$, $1 \leq i \leq N$, the distribution of the embedded process $\{I_n\}_{n>0}$, is readily available since it holds that $\pi_i = \sum_{j=ik}^{(i+1)k-1} \tilde{\pi}_j$. The $1 \times Nk$ steady-state vector $\tilde{\pi}$ is obtained as follows. Define the $Nk \times Nk$ matrix $Q$ as $P - I$, where $I$ is the unity matrix of dimension $Nk$. The matrix $Q$ can be seen as an infinitesimal generator of a continuous time Markov chain and $Q$ can be written in a lower block-Hessenberg form by relabeling the states appropriately. Therefore, we can calculate the unique stochastic vector $\tilde{\pi}$ for which $\tilde{\pi}Q = 0$, i.e., $\tilde{\pi}P = \tilde{\pi}$, in an efficient manner using the Latouche-Jacobs-Gaver algorithm [20], which has a time complexity of $O(k^3N^2)$ and a space complexity of $O(k^2N)$.

## 4.2 Intrastream synchronization (continuity) metrics

Expanding or shortening the duration of a frame presentation, as done by the playout policies of equation (1) or any other playout policy, introduces a discontinuity – a loss of intrastream synchronization – quantified by the difference between the selected frame duration and the normal frame duration $T$. Let $d_i(D_i)$ denote the *discontinuity* that is incurred when the next frame (frame $n$) is presented with a duration $D_i$ and the current phase occupancy is $\{\tilde{I}_n\} = i$.

$$d_i(D_i) = |D_i - T + S_i(D_i)| \tag{3}$$

The term $D_i - T$ quantifies the discontinuity incurred by choosing a playout duration other than the normal one, $T$. In addition, the metric has to cater for a possible underflow that might follow the completion of the nominal duration $D_i$. Such an underflow would occur if the buffer is found without a complete frame, $D_i$ time units after the initiation of the $n$th presentation. In that occassion frame $n$ will remain on the display for an additional interval $S_i(D_i)$ until the next frame becomes available. This brings up its duration to $D_i + S_i(D_i)$ (see Fig. 4). The absolute value is used in (3) because the quantity $D_i - T + S_i(D_i)$ may assume negative values. Such is the case when $D_i < T$ and no underflow occurs. Also note that a truncated duration and an underflow may cancel each other. Thus a truncated duration, $D_i = T/2$, followed by an underflow with duration $T/2$ inadvertently lead to a normal presentation duration and no discontinuity.

The underflow interval $S_i(D_i)$ depends on the current buffer occupancy $i$, the selected presentation duration $D_i$, and also the number of new phase arrivals over $D_i$, $y$. Under $k$-Erlang arrivals:

$$E\{S_i(D_i)|y\} = \begin{cases} (k - (i - k + y)) \cdot T/k & , i - k + y < k \\ 0 & , i - k + y \geq k \end{cases} \tag{4}$$

In the first case the system is left with $i'$ phases at $t_n + D_i$ which amount to less than a complete frame, thus an additional $k - i'$ phases must arrive. The expected time for that is $(k - i') \cdot T/k$. In the second case the buffer is non-empty at $t_n + D_i$ thus no underflow occurs and the next frame $(n+1)$ is displayed exactly $D_i$ time units after the initiation of frame $n$.

We define a more generalized continuity metric – called the Distortion of Playout metric (abbreviated DoP) – which includes the notion of discontinuity as in (3) but also accounts for any lack of continuity due to buffer overflows over the current presentation interval.

$$DoP_i(D_i) = d_i(D_i) + L_i(D_i) \cdot T, \tag{5}$$

where $L_i(D_i)$ is a random variable that denotes the number of newly arrived frames that are dropped/lost, due to buffer overflows, over the current frame presentation duration $D_i$, given that the number of phases in the system was $i$ at $t_n$:

$$P[L_i(D_i) = x] = \begin{cases} \sum_{j=0}^{k-1} P\{(N+x)k - i + j, D_i\} & , x > 0 \\ \sum_{j<(N+1)k-i} P\{j, D_i\} & , x = 0 \end{cases} \tag{6}$$

Note that the DoP metric measures time; it adds the duration of all time intervals during which the smooth playout of frames is disrupted. A basic idea reflected in the definitions of both $d_i(D_i)$ and $DoP_i(D_i)$ is that the perceptual cost of an idle time gap between two frames (occurring when the first frame stays on display for more than $T$) is equal to the perceptual cost of a loss-of-information discontinuity (an overflow or a fast-forward) of equal duration. This is based on recent perceptual studies [21] where it is shown that jitter degrades the perceptual quality of video nearly as much as packet loss does. For an example in support of this claim, we may think that an underflow with a duration $T$, degrades stream continuity, nearly as much, as does a lost frame.

It must also be noted that there is a delicate semantic difference between overflow disruptions and all other cases of disruption (underflows, modified playout durations). The latter are immediately experienced during the presentation of frame $n$ whereas an overflow, although occurring during the presentation of frame $n$, is experienced in the future (when the playout skips one or more overflowed frames). With the current formulation (equation (5)) the disruption caused by any overflows during the presentation of frame $n$ is added to the overall disruption of this frame instead of a future one. This is required in order to preserve the tractability of the model. Otherwise various states should be introduced, adding the required memory that allows for the association of past overflows to the currently presented frame.

# 5 Derivation of the optimal playout scheduler for different levels of network jitter

In this section we develop a Markov decision model which leads to the derivation of the optimal playout scheduler for different levels of network jitter, as captured by the various $k$ values of the assumed $k$-Erlang arrival process.

## 5.1 The Markov decision process

In Section 4 the Markov chain $\{\tilde{I}_n\}_{n>0}$ was used in the context of the $E_k/D_i/1/N$ queue for the derivation of the steady-state behavior for a given – occupancy dependent – playout policy $(D_i)$. In this section, $\{\tilde{I}_n\}_{n>0}$ is generalized into a discrete time Markov decision process (MDP) with the aim of deriving the optimal playout policy for different jitter levels (captured by the Erlang parameter $k$). Let $\{\tilde{I}^*_n\}_{n>0}$ be the MDP obtained from the $\{\tilde{I}_n\}_{n>0}$ Markov chain by adding an action following each observation instant (at the time prior to the next playout). This action explicitly defines the playout duration for the next frame and by doing so it incurs an immediate cost and also affects the probability law for the next transition. For the formal definition of the MDP one needs to define a tuple $\langle \mathcal{S}, \mathcal{A}, P, C \rangle$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of possible actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function specifying the probability $P\{j|i, a\} \equiv t_{ij}(a)$ of observing a transition to state $j \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $i \in \mathcal{S}$ and, finally, $C : \mathcal{S} \times \mathcal{A} \to \Re$ is a function specifying the cost $c_i(a)$ of taking action $a \in \mathcal{A}$ at state $i \in \mathcal{S}$.

The state space $\mathcal{S}$ of $\{\tilde{I}^*_n\}$ comprises all possible phase occupancy levels, thus takes values in $[k, (N+1)k-1]$. An *action* is defined to be the choice of an integer value $a$ that explicitly determines $B(a)$, the presentation duration for the next frame through:

$$B(a) = T \cdot \frac{a}{\alpha} \tag{7}$$

$\alpha$ defines the basic adjustment quantum which is equal to $T/\alpha$. The action space for the problem is $\mathcal{A} = [1, M]$, where $M$ is an integer value that results in the maximum allowable playout duration $B(M)$. Notice that in (7) when $a > \alpha$ ($a < \alpha$) the resulting playout duration is larger (smaller) than the normal frame duration. When altered playout durations (larger/smaller) are applied to a series of consecutive frames, they constitute a transient alternation of the playout rate (an effect that resembles a slowdown/fast-forward operation in a VCR). The transition probabilities of the MDP are "decision-dependent"; they are described by equation (2), but with service durations that are not a priori known, as in the case of a known state dependent service, but depend on the chosen action, i.e., $t_{ij}(a) \equiv p_{ij}(B(a))$.

The goal of the decision model is to prescribe a playout *policy* – a rule for choosing the duration of the next frame based on the current state. In the general case a policy $R \equiv \{D_{ia} : i \in \mathcal{S}, a \in \mathcal{A}\}$ is a mapping: $\mathcal{S} \times \mathcal{A} \to [0, 1]$; it is completely defined for a given tuple $\langle \mathcal{S}, \mathcal{A}, P, C \rangle$ by the probabilities: $D_{ia} \triangleq P\{\text{action} = a|\text{state} = i\}$. The derived optimal policy, under the considered minimization objective and the selected solution method (described in Appendix A), always prescribes the same action whenever at the same state, i.e., the optimal policy is non-randomized (see [22] for details). Under a non-randomized policy the probabilities $D_{ia}$ are either 0 or 1. In view of this observation, the exact definition of the non-randomized policy $R$ reduces to the definition of the function $A_i(R) = a$ which for every $i \in \mathcal{S}$ returns the selected action $a$.

As mentioned earlier, $c_i(a)$ denotes the cost incurred when action $a$ is taken when the phase occupancy process is in state $i$. The optimal policy $R_{opt}$ is defined to be the policy that minimizes $E_R\{c\}$, where $E_R\{c\}$ denotes the long-run average cost induced under some policy $R$. If $\tilde{\pi}_i(R)$ denotes the steady state probability that the Markov chain $\{\tilde{I}_n\}$, with $D_i = (A_i(R)/\alpha)T$, is in state

$i$, then

$$R_{opt} = \arg\min_R E_R\{c\} \quad \text{with} \quad E_R\{c\} = \sum_{i=k}^{(N+1)k-1} c_i(A_i(R)) \cdot \tilde{\pi}_i(R) \tag{8}$$

A number of techniques are known for the derivation of the optimal policy of (8); these include exhaustive enumeration (only for small systems), linear programming, policy-iteration, and value iteration. The current MDP problem was solved by using a value-iteration algorithm (described in Appendix A) which takes as input the action-dependent transition probabilities, $t_{ij}(a) = p_{ij}(B(a))$, and the state-action costs, $c_i(a)$ (defined in detail in 5.2), and returns $R_{opt}$.

## 5.2 Cost assignment

An appropriate MDP cost is described in this section; it "penalizes" the *lack of continuity* that may arise from a certain action. This lack of continuity may be directly experienced as in the case of a frame presentation with a duration smaller or larger than $T$. In addition, the continuity cost also accounts for any lack of continuity due to overflowed (lost) frames occurring during that interval.

A candidate for the continuity cost is: $DoP_i(a) = E\{DoP_i(B(a))\}$, i.e., the expected value of $DoP_i(D_i)$ with respect to the number of new arrivals, with $D_i = B(a)$ (see definition of $DoP_i(D_i)$ in (5)). $DoP_i(a)$ is a legitimate MDP cost as it depends only on the current state $i$, and the selected action $a$. Such a cost assignment returns an $E\{DoP\}$-optimal policy, i.e., a policy that minimizes: $E_{\mathcal{S}}\{DoP\} = \sum_{i \in \mathcal{S}} \tilde{\pi}_i(R) \cdot DoP_i(A_i(R))$ over all the policies $R$ defined in $\mathcal{S} \times \mathcal{A}$. The results presented in [5] show that $R_T$, the static deterministic policy with constant presentation durations equal to the frame period, is $E\{DoP\}$-optimal under Poisson frame arrivals. The same applies also to the case of $k$-Erlang arrivals (see the results of Sect. 6).

The minimization of $E_{\mathcal{S}}\{DoP\}$ calls for the minimization of the average amount of synchronization loss which is due to: underflow discontinuities, slowdown discontinuities, overflow discontinuities and fast-forward discontinuities. The minimization of $E_{\mathcal{S}}\{DoP\}$ is a rightful objective but cannot guarantee the perceptual optimality, as it only caters to the minimization of the average loss of synchronization, without paying any attention as to how this loss of synchronization spreads in time. It has been realized that the human perceptual system is more sensitive to a small frequency of long-lasting disruptions than to a higher frequency of short-lived disruptions [3]. This is due to 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 (slowdowns and fast-forwards), even when the latter lead to a higher value for $E_{\mathcal{S}}\{DoP\}$. Thus, a playout policy should be allowed to increase $E_{\mathcal{S}}\{DoP\}$ if this increase provides for a smoother spacing between synchronization-loss occurrences, thus help in concealing them. We pursue this idea by defining the state-action cost to be:

$$c_i(a) = \beta \cdot DoP_i(a) + (1 - \beta) \cdot DoP_i^{(2)}(a) \tag{9}$$

where $DoP_i^{(2)}(a) = E\{(DoP_i(B(a)))^2\}$ is the expected square value of $DoP_i(B(a))$ with respect to the number of new arrivals. The weighing factor $\beta$ is a user-defined input that controls the relative importance between the two minimization objectives: the minimization of $E_{\mathcal{S}}\{DoP\}$, and the minimization of $E_{\mathcal{S}}\{DoP^2\}$[5]. Setting $\beta = 1$ leads to the minimization of $E_{\mathcal{S}}\{DoP\}$ without any regard for the variability of the duration of synchronization loss occurrences. Setting $\beta = 0$ leads to the minimization of $E_{\mathcal{S}}\{DoP^2\}$ and the resulting $E_{\mathcal{S}}\{DoP^2\}$-optimal policy induces smoother synchronization losses than the $E_{\mathcal{S}}\{DoP\}$-optimal policy. As it will be shown later, the reduction of $E_{\mathcal{S}}\{DoP\}$ comes at the cost of an increased $E_{\mathcal{S}}\{DoP^2\}$ and vice versa. Values of $\beta$ that fall between the two extremes (0 and 1) provide various levels of compromise between $min\{E_{\mathcal{S}}\{DoP\}\}$ and $min\{E_{\mathcal{S}}\{DoP^2\}\}$. The optimality of this tradeoff stems from the fact that for a given value of one of the continuity components, the derived optimal solution will provide a minimal value for the other continuity component. In essence, the designer of the PVR selects a $\beta$ that results in a desired value of $E_{\mathcal{S}}\{DoP^2\}$ ($E_{\mathcal{S}}\{DoP\}$) and knows that for that value of $E_{\mathcal{S}}\{DoP^2\}$ ($E_{\mathcal{S}}\{DoP\}$) there cannot exist a policy that provides a smaller $E_{\mathcal{S}}\{DoP\}$ ($E_{\mathcal{S}}\{DoP^2\}$) than the $E_{\mathcal{S}}\{DoP\}$ ($E_{\mathcal{S}}\{DoP^2\}$) provided by the proposed playout policy (since that policy minimizes a cost expression that involves both $E_{\mathcal{S}}\{DoP^2\}$, $E_{\mathcal{S}}\{DoP\}$).

As a final comment on the employed cost we note that although lossed are assigned to the ongoing presentation instead of a future one (as discussed in Sect. 4.2) $E_{\mathcal{S}}\{DoP\}$ and $E_{\mathcal{S}}\{DoP^2\}$ represented meaningful continuity metrics. It can be easily shown that $E_{\mathcal{S}}\{DoP\}$ is immune to the mapping of losses to frames; it only depends on the number of losses. On the other hand, $E_{\mathcal{S}}\{DoP^2\}$ is affected by the exact mapping of losses to frames due to the existence of the square power. However, due to the fact that overflow continuity disruptions are generally much larger (especially when batch losses occur) than all other kinds of disruptions that may occur under high occupancies, the difference between the employed approach and the more accurate, but complicated one, is rather marginal.

# 6 Numerical results and discussion

In this section we apply the developed optimization model to derive the optimal playout policy as defined in (8). In all the examples the duration of a frame will be equal to 33 msec (implying 30 frames/sec). Unless stated otherwise, the playout buffer capacity $N$ will be equal to 30. Two granularities are used for the adjustment of frame durations: 3.3 msec (corresponding to $\alpha = 10$), and 1 msec (corresponding to $\alpha = 33$).

## 6.1 Optimal playout policies for different levels of network jitter

The continuity weight $\beta$ was introduced in equation (9) for the regulation of the relative importance between the mean value and the variability of $DoP_i(a)$. In [5] we assumed Poisson arrivals and showed that by letting $\beta$ take values in $[0, 1]$ we can achieve various tradeoffs between the minimization of the average DoP and its variability. For $\beta = 1$ the derived optimal playout policy mandated that all frames be played at their normal duration, that is, the deterministic service (DS) was shown to be $E\{DoP\}$-optimal. For $\beta = 0$ we obtained the $E\{DoP^2\}$-optimal policy which applied a considerable amount of playout regulation towards the two extremes of the occupancy of the playout buffer.

[5]We are referring to the minimization of $\sum_{i \in \mathcal{S}} \tilde{\pi}_i(R) \cdot DoP_i^{(2)}(A_i(R))$ over all the policies $R$ defined in $\mathcal{S} \times \mathcal{A}$.

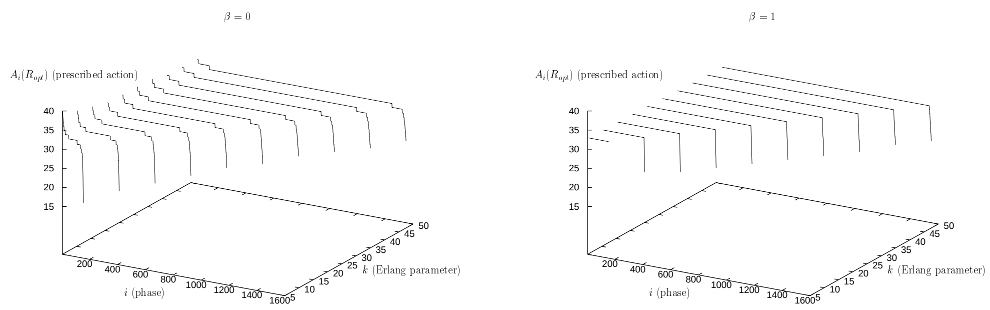


Figure 5: The prescribed optimal action $A_i(R_{opt})$ (on the y-axis) for each phase occupancy (on the x-axis), for the cases of $\beta = 0$ and $\beta = 1$ and different $k$ (on the z-axis). The normal frame duration is obtained with $A_i(R_{opt}) = 33$, i.e., the adjustment granularity is $\alpha = 33$. The playout buffer has a capacity for $N = 30$ frames.

The aforementioned behavior generally applies also to the policies derived for $k$-Erlang arrivals, nevertheless, the amount of playout regulation required by a given $\beta$ is affected by the Erlang parameter $k$. For increased network jitter (small $k$), the derived policies apply an increased amount of playout regulation, eventually reaching the behavior under Poisson for $k = 1$. As frame arrivals become more regular (with larger $k$), the amount of playout regulation required to minimize the average cost, for a selected $\beta$, decreases. This is on a par with the intuitive guess that playout manipulation ought to "smooth-out" as jitter drops.

Figure 5 shows the structure of the derived playout policies for $\beta = 0$ and $\beta = 1$ and for different levels of delay jitter. Due to the fact that $k$ affects the state-space of the Markov decision problem, the number of phases (on the x-axis), grows with $k$ (on the z-axis) despite that all systems have the same playout buffer[6] ($N = 30$). The left graph illustrates the structure of playout policies for $\beta = 0$. This produces policies that minimize $E_{\mathcal{S}}\{DoP^2\}$ by applying a substantial amount of playout regulation at buffer extremes. As it may be seen, the amount of playout regulation reduces with $k$ (note how the prescribed actions, at the edges of the buffer, tend to be less severe with increasing $k$.). The right graph corresponds to $\beta = 1$ which leads to policies that minimize $E_{\mathcal{S}}\{DoP\}$. Such policies apply almost no playout regulation, except in the last few phases (which however have a very small probability of being visited under steady state), thus are approximated very closely by the deterministic service that presents all frames at their normal playout duration. Intermediate values, $0 < \beta < 1$, lead to policies that fall between the two extreme cases. In the sequel we will be focusing in $\beta$ confined in the initial range $0 < \beta < 0.1$. Notice that the two components that contribute to the $c_i(a)$ cost of equation (9) have different units thus only small values of $\beta$ lead to a balanced tradeoff between the two cost components. Values of $\beta$ larger than 0.1 turn almost entirely in favor of minimizing $E_{\mathcal{S}}\{DoP\}$ thus amount almost to the presented behaviour for $\beta = 1$.

The two plots of Fig. 6 illustrate the performance results of the derived Erlang-optimal (EO) policies. For a particular policy, $R(\beta, k)$[7], both $E_{\mathcal{S}}\{DoP\}$ and $E_{\mathcal{S}}\{DoP^2\}$ improve (reduce) with the regularity of arrivals (that is, with $k$). Given a jitter level $k$, a small $\beta$ favors the reduction of $E_{\mathcal{S}}\{DoP^2\}$, while a large $\beta$ favors the reduction of $E_{\mathcal{S}}\{DoP\}$. Figure 7 is identical to Fig. 6 but

[6] For $k = 5$ a total of $Nk = 150$ phases are required to fill the playout buffer with 30 frames, while for $k = 10$ the required number of phases is 300.

[7] E.g., $R(0.2, 10)$ denotes the optimal policy if we set $\beta = 0.2$ and $k = 10$.

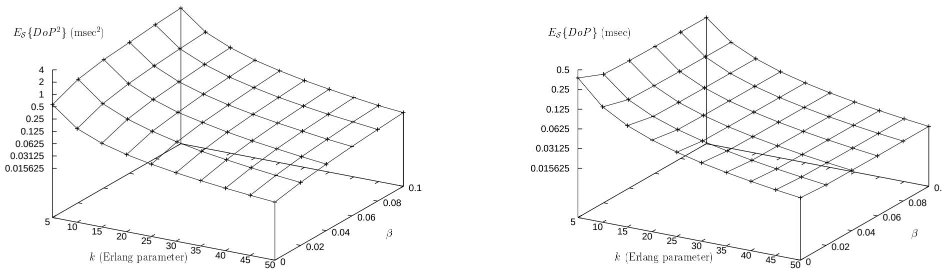


Figure 6: The $E_{\mathcal{S}}\{DoP^2\}$, $E_{\mathcal{S}}\{DoP\}$ performance of EO policies for different values of the continuity weight $\beta$ and for different $k$ ($\alpha = 33, N = 30$).

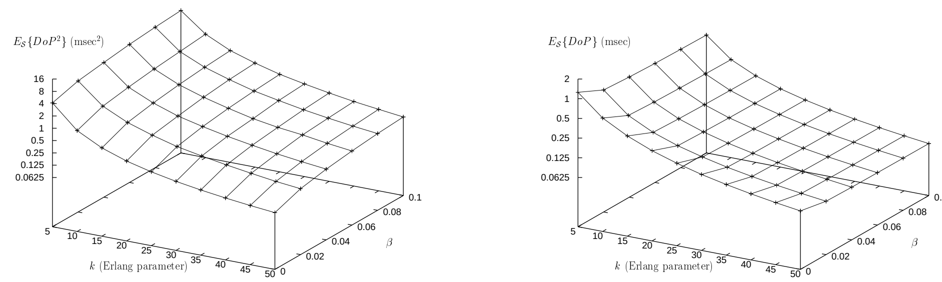


Figure 7: The $E_{\mathcal{S}}\{DoP^2\}$, $E_{\mathcal{S}}\{DoP\}$ performance of EO policies for different values of the continuity weight $\beta$ and for different $k$ ($\alpha = 33, N = 10$).

corresponds to a smaller buffer, $N = 10$; notice that both continuity metrics deteriorate as a result of the smaller offered buffer capacity.

The selected range for $k$ has been limited to values up to $k = 50$ stimulated mainly by the measurement experiments. An important question is whether the optimization model is "solvable" for larger $k$ values in a "practical" time. We have solved the model for $k$ up to 150 in a reasonable time (for an off-line computation). Larger $k$'s are probably not necesary to consider for the following two reasons: (1) the measured interarrival traces were not that regular; (2) most significantly, for such a small delay jitter it is not necessary to derive the EO policy – its sufficient to use the deterministic service (DS). The latter observation stems from the fact that the amount of playout regulation reduces with delay jitter (see Fig. 5, as well as Fig. 10 in the sequel) and thus eventually the best playout policy is to play all frames at their normal duration.

## 6.2 Controlling the buffering delay

The proposed Erlang-optimal (EO) playout policies can be configured to suit the delay requirements of different streaming applications. The simplest way to do so is by limiting the size of the playout buffer. The maximum buffering delay of a presented frame is approximately equal to $N \cdot T$, since the average playout duration applied by EO policies is approximately equal to $T$. It is known [1]

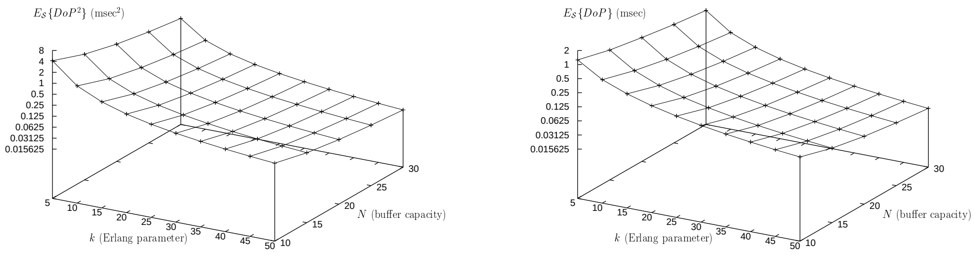


Figure 8: The $E_{\mathcal{S}}\{DoP^2\}$, $E_{\mathcal{S}}\{DoP\}$ performance of EO policies for different values of the buffer capacity $N$ and for different $k$ ($\alpha = 33, \beta = 0$).

that a maximum network jitter[8], $j_{max}$, can be completely removed by accumulating a playout buffer of equal duration. With the playout buffer for 30 frames and a frame rate of 30 frames/sec, the system can completely eliminate the jitter, if this jitter is confined below $N \cdot T = 1$ sec. The jitter, however, can be larger, especially if we also include the server-induced jitter (e.g., in an overloaded video-on-demand server); also the delay requirements of the application could require a smaller maximum buffering delay (e.g., up to 330 msec, achieved with $N = 10$). In such cases underflows and overflows might occur (since $j_{max} > N \cdot T$) and the EO playout policy will try to conceal them by manipulating the duration of frames at the extremes of the buffer.

The inherent tradeoff between continuity-quality and delay also applies to the proposed EO policies. When the (mean/maximum) buffering delay decreases, e.g., because a smaller playout buffer is available, the overall stream continuity (both components, $E_{\mathcal{S}}\{DoP\}$ and $E_{\mathcal{S}}\{DoP^2\}$) deteriorates. The two plots of Fig. 8 show that for a given $k$, a smaller buffer capacity $N$ always provides a worse $E_{\mathcal{S}}\{DoP\}$, $E_{\mathcal{S}}\{DoP^2\}$.

A more sophisticated delay control method has been presented in [5] by associating an appropriate delay cost to the state-action cost of equation (9). Under this model, the playout policy in some cases applies an increased playout rate with the aim to reduce the occupancy of the buffer and, hence, the buffering delay as well. This method is readily applicable to the EO policy as well.

### 6.3 Comparison of playout policies: Erlang-optimal vs. Poisson-optimal and deterministic service

In this section a comparative study of the performance of the Erlang-optimal (EO), the Poisson-optimal (PO) and the deterministic service (DS) policies is presented. It is assumed that the true interarrival process is i.i.d. and $k$-Erlang distributed. As it is clear for earlier discussions, such an arrival process may induce a wide range of delay jitter, from no jitter ($k = \infty$, deterministic arrivals) to the (high) level corresponding to Poisson arrivals ($k = 1$). Thus, a wide range of real network delay jitter (second moment of interarrivals) can be matched with that under a $k$-Erlang distribution for some $k$.

Since the interarrival process is i.i.d. and $k$-Erlang distributed, the EO policy will lead to the

[8]The maximum network jitter is defined as the difference between the largest and the smallest network delay, i.e., $j_{max} = \max_i D_{n,i} - \min_i D_{n,i}$.

best possible performance, as this policy is the one optimized for such an interarrival distribution. When the PO policy is employed when the interarrivals are i.i.d. and $k$-Erlang distributed the resulting performance cannot outperform that under the EO policy. The same holds true if the DS policy is employed when the interarrivals are i.i.d. and $k$-Erlang distributed.

The general observation is that, when the frame arrivals are more regular than Poisson, PO policies (for different continuity weights $\beta$ and/or adjustment granularities $\alpha$) become suboptimal since they apply an excessive amount of playout regulation. In such cases, the corresponding EO policies provide an overall improved performance by applying an appropriate amount of playout regulation. The DS service provides for the minimal $E_{\mathcal{S}}\{DoP\}$, but has a large $E_{\mathcal{S}}\{DoP^2\}$, especially with small $k$. For the purpose of comparison, normalized, rather than absolute performance metrics will be presented. The normalization is carried out by dividing the $E_{\mathcal{S}}\{DoP^2\}$ ($E_{\mathcal{S}}\{DoP\}$) performance of an EO or PO policy with the performance of DS for the same $k$. A value smaller (larger) than 1 means that the corresponding policy performs better (worse) than DS for the involved metric. This normalization schemes leads to a subjective evaluation of policies which quantifies how much better or worse they would perform as compared to DS, under the same level of delay jitter.

Figure 9 illustrates the normalized performance, in terms of $E_{\mathcal{S}}\{DoP^2\}$ and $E_{\mathcal{S}}\{DoP\}$, of the following playout policies: the $E\{DoP^2\}$-optimal [9] ($\beta = 0$) PO($\alpha$), the $E\{DoP^2\}$-optimal ($\beta = 0$) EO($\alpha$). The plots shows that DS is constantly inferior in terms of the variability of discontinuity occurrences to both EO and PO. EO guarantees a lower $E_{\mathcal{S}}\{DoP^2\}$ compared to the corresponding PO of the same $\alpha$, for all $k$ in the presented range (compare the couples EO(10)-PO(10) and EO(33)-PO(33)). By increasing the granularity of playout manipulation both EO and PO improve significantly; PO(33) is better than PO(10) and EO(33) is better than EO(10). Notice that $\alpha = 33$ leads to an adjustment quantum of 1 msec, which is equal to the best timing granularity that most general operating systems support. A larger $\alpha$ would lead to a smaller granularity and an improved performance but this performance would only be a theoretical one, since the corresponding policy would not be implementable in practice. Thus the presented results for $\alpha = 33$ should be viewed as the optimal attainable ones.

DS incurs long-lasting discontinuities by not applying any playout regulation at the buffer extremes, but achieves the best $E_{\mathcal{S}}\{DoP\}$ (see, the right graph of Fig. 9). EO(10) and EO(33) follow closely, while PO(10) and P(33) are much worse.

In conclusion, the following should be noted: (1) there is a *consistent* performance tradeoff between DS and EO, with the former (latter) providing a superior $E_{\mathcal{S}}\{DoP\}$ ($E_{\mathcal{S}}\{DoP^2\}$), across all $k$; (2) EO clearly outperforms PO since it provides a better performance with respect to both components of continuity, $E_{\mathcal{S}}\{DoP\}$ and $E_{\mathcal{S}}\{DoP^2\}$. Referring back to Fig. 9 it may be seen that EO(33) provides for a large reduction of variability of discontinuities. For most $k$, an $E_{\mathcal{S}}\{DoP^2\}$ that is only 6% of the corresponding value of DS is achieved (the corresponding PO is limited to around 40-50% of DS). The cost for providing this large reduction of variability is a small increase by a factor of 1.02 (2% worse) of $E_{\mathcal{S}}\{DoP\}$ as compared to DS (here PO performs poorly, as much as 8 times worse than DS). These results indicate that EO(33) could provide a significantly improved perceptual quality as compared to all other policies. Finally, although omitted for brevity, it can been shown that EO is much better than the threshold slowdown (TS) policy (in [5] we have

[9] This is a policy that minimizes the variability of DoP under Poisson arrivals. It has been introduced in [5].

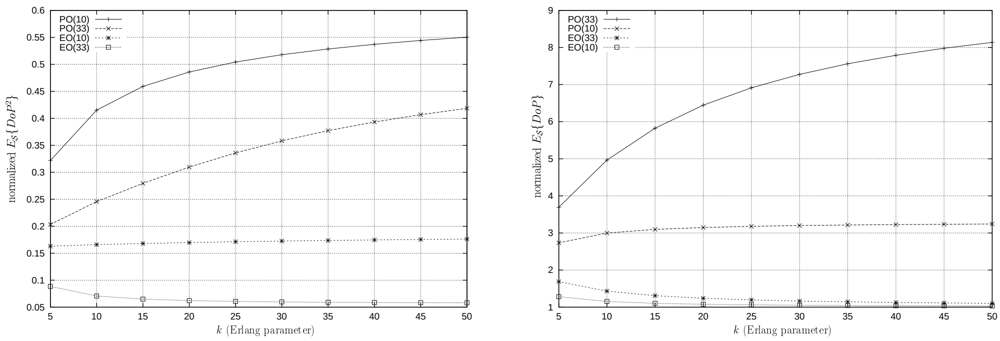


Figure 9: The normalized $E_{\mathcal{S}}\{DoP^2\}$ and $E_{\mathcal{S}}\{DoP\}$ performance of different playout policies with respect to DS, for different values of $k$: PO(10), PO(33), EO(10), EO(33). The Erlang and Poisson policies are $E_{\mathcal{S}}\{DoP^2\}$-optimal, i.e., they have been optimized for the minimization of variability of DoP by selecting $\beta = 0$. A playout buffer for 30 frames is used in all the examples.

compared PO and TS).

# 7 Applying the phase-aware optimal policy to a real-world video receiver

The obtained EO playout policies indicate that the amount of playout regulation required to obtain the optimal performance varies with the regularity, that is, variation, of the frame arrival process. This observation suggests that an implemented receiver should also vary the extent of playout regulation with varying network jitter.

To apply the analytically derived optimal playout policy of Sect. 5 to a real-world PVR, the implemented version of the derived optimal playout scheduler must be aware of the total number of phases in the system upon a presentation completion instant (say of the $n$th frame). In the aforementioned analytical model the phase occupancy process $\{\tilde{I}_n\}$ provided that exact information by accounting for both the phases corresponding to the number of frames already in the buffer ($kI_n$ phases), as well as the phases completed by the ongoing arrival ($J_n$ phases). In an implemented system $\{\tilde{I}_n\}$ is only *partially observable*, since only one of its components is directly measurable – $kI_n$, the number of phases that correspond to the buffered frames. $J_n$, the second component that contributes to $\{\tilde{I}_n\}$, is unknown.

An estimator could be designed to carry-out the estimation of $J_n$ and use the obtained result to select the action that corresponds to $kI_n$ phases plus the estimation of $J_n$. An estimate $\hat{J}_n(E_n)$ can be derived by using the *last-arrival elapsed time* $E_n$, which is the interval from the last frame arrival ($a_n$) up to the present time of completion of the $n$th playout ($t_n$) (see Fig. 3). The scheduler logs the time of the last frame arrival and uses it to measure $E_n$. Knowing $E_n$, the expected number of completed phases over that interval can be computed, thus providing the estimate $\hat{J}_n(E_n)$. This method has been evaluated by simulation and has presented moderate success. Its performance, however, deviates significantly from the theoretical performance when $k$ is large ($k > 35$). Additionally, the incorporation of the estimator adds to the complexity at the receiver.

We have derived a simple method that can overcome the problems associated with the estimation

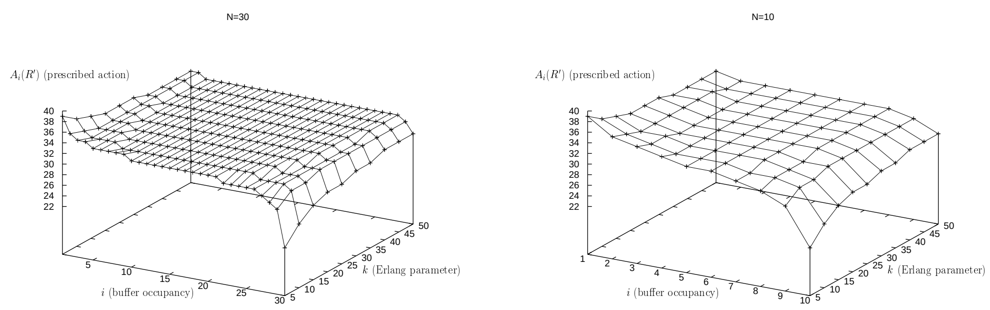


Figure 10: The structure of "collapsed" EO policies for different jitter levels, $1 \leq k \leq 50$. The two plots correspond to different buffer capacities: $N = 30$ and $N = 10$ ($\alpha = 33, \beta = 0$).

of the phase. This methods provides a performance that is very close to the theoretical optimal and is always better than the simulation results obtained for the estimator. We approximate the theoretical phase-aware optimal policy, $R_{opt}$, by an appropriate phase-unaware[10] policy, $R'_{opt}$, that introduces an *equivalent* amount of playout regulation (at buffer extremes) resulting in a *similar* playout quality. Our results show that a performance very close to the theoretical optimal can be achieved this way.

The transformation of the phase-aware policy $R_{opt}$, characterized by the function $A_i(R_{opt})$, for $k \leq i \leq (N+1)k-1$, to the phase-unaware policy $R'_{opt}$, characterized by the function $A_i^u(R'_{opt})$, for $1 \leq i \leq N$, is carried out by averaging over the suggested actions for each $k$-tuple of phases, corresponding to a single frame occupancy, and use the averaged action for that frame occupancy after rounding it to the closest integer.

$$A_i^u(R'_{opt}) = round\left(1/k \cdot \sum_{j=i \cdot k}^{(i+1) \cdot k-1} A_j(R_{opt})\right) \quad ,1 \leq i \leq N \tag{10}$$

Figure 10 depicts the structure of $R'_{opt}$ for $\beta = 0$ and different $k$, for two buffer capacities. These plots correspond to phase-unaware versions of the phase-aware policy illustrated in Fig. 5. They indicate that for the minimization of $E_{\mathcal{S}}\{DoP^2\}$, the playout manipulation ought to "smooth" with $k$.

Figure 11 illustrates that a real-world video receiver, equipped with an estimator to determine the amount of network jitter, could significantly reduce the variation of the DoP metric by applying the phase unaware version of the optimal policies, all of which were determined off-line. Note that the lines that correspond to EO(33) and CEO(33) (collapsed EO) policies almost coincide over all $k$; the theoretical EO(33) and the applied CEO(33) policies essentially provide for the same performance, which is always much better than that under the PO and DS policies.

An important question is whether a reduced $E_{\mathcal{S}}\{DoP\}$, $E_{\mathcal{S}}\{DoP^2\}$ is really equivalent to a significantly better visual perception. This is a matter that ought to be determined by future experiments. Even if it turns out that this is not always the case, it is possible to apply the techniques presented in this paper to obtain optimal policies with respect to other, perhaps more

[10]We refer to a policy $R$ as phase-unaware, if the playout duration depends solely on the number of frames in the buffer, $I_n$, that is, it is independent of $J_n$. Otherwise, it is referred to as a phase-aware policy. Therefore, we can characterize a phase-unaware policy $R$ by a function $A_i^u(R)$, with $1 \leq i \leq N$.

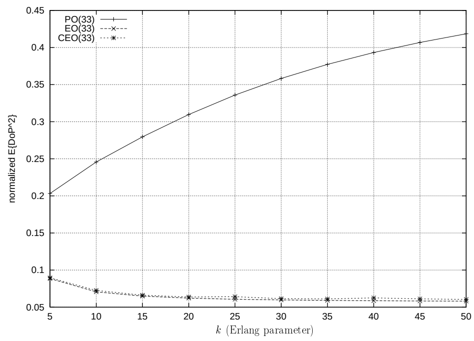


Figure 11: The normalized $E_S\{DoP^2\}$ performance with respect to DS for PO(33), EO(33) and CEO(33) (collapsed EO) for different levels of delay jitter, $1 \leq k \leq 50$ ($\alpha = 33, \beta = 0$).

suitable, (dis)continuity metrics.

# 8 Overall system architecture

In this section we describe some scenarios for the exploitation of the developed playout policies in implemented PVRs.

The obtained theoretical EO policies, and their corresponding implementable CEO policies, are optimized for a given level of network jitter, captured by the Erlang parameter $k$. It is known, however, that the jitter in a best-effort network fluctuates over various time scales and is highly affected by a plethora of parameters such as: the co-existing traffic mix, the underlying network technology, the distance between the communicating end-points etc. Two time scales of particular interest are: (1) the transient increases of jitter in small time periods; (2) the somewhat permanent jitter, that relates to increased load in the network, e.g., during the "peak hour" of operation.

Most playout schedulers in the past literature [1] try to monitor the current level of network jitter and adjust accordingly. When the network jitter is stable, most systems are able to conceal it by applying an appropriate playout algorithm. A phenomenon that complicates the operation of a scheduler is the existence of sharp "delay spikes" (sudden increases of the network delay of some frames) which have been reported by various studies [23, 24]; they are attributed to causes such as: surges of peak traffic, administrative functions in routers that result in the distraction of the CPU from packet forwarding etc. These delay spikes seriously degrade the presentation quality, in a PVR that otherwise can be thought as being in a "steady-state" (having buffered enough frames to cope with the average jitter). Some systems for packet voice communications [25, 26] (which make use of timing information) operate in a dual mode, with the second mode of operation being devoted to the detection and the concealment of delay spikes. The normal mode of operation uses an estimator, which appreciates the current level of jitter, without having to be very "reactive", since jitter is assumed to drift rather slowly.

Our EO policies can be thought as an analogous way of handling the delay variability, with a buffer-oriented scheduler, destined for packet video communications. Under a given average jitter (reflected in $k$), a delay-spike is handled by the playout regulation towards the buffer extremes (avoiding long-lasting discontinuities: underflows and overflows). These delay spikes are modeled

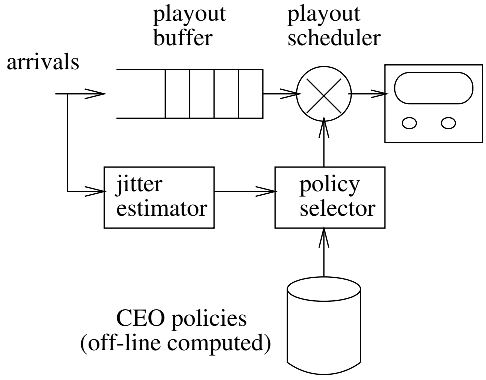


Figure 12: Block diagram of an implemented system.

by the occasional peaks of the $k$-Erlang, moreover, the produced behavior with different $k$ maps well to the real environment: small $k$ produce more peaks, as is the case when the network is congested, while larger $k$ result in infrequent peaks. On a larger time scale, e.g., oriented toward the *time-of-day* load behavior, a PVR can switch to an appropriate EO policy for that jitter level.

Figure 12 depicts the envisaged implementation; a PVR estimates the current level of network jitter by observing the frame interarrivals and "loads" the appropriate, off-line computed CEO playout policy. The estimation of the average network jitter can be performed by a standard linear recursive estimator [27] that estimates the variance of interarrivals, $V_i$, at the $i$th playout by using the corresponding interarrival times, $X_i$:

$$\begin{aligned} \hat{X}_i &= g \cdot \hat{X}_{i-1} + (1-g) \cdot X_i \\ \hat{V}_i &= h \cdot \hat{V}_{i-1} + (1-h) \cdot (\hat{X}_{i-1} - X_i)^2 \end{aligned} \tag{11}$$

The estimates $\hat{X}_i$, $\hat{V}_i$ may be used for the selection of the appropriate $k$ that corresponds to the corrently observed jitter. This can be done by using the relationship $Var\{Y\}/E^2\{Y\} = 1/k$ which holds true for a $k$-Erlang distributed random variable $Y$. Thus an estimate $\hat{k}$ is obtained by using $\hat{k} = round((\hat{X}_i)^2/\hat{V}_i)$. The CEO that corresponds to $\hat{k}$ is loaded from the repository of precomputed playout policies. By letting $g, h$, in (11) assume large values (close to 1) the estimate $\hat{k}$ remains stable, in the sense that transient delay-spikes are filtered-out, not causing unnecessary changes of playout policy. Only permanent changes of jitter are allowed to pass the filter and trigger the exchange of playout policy. Estimators such as (11) have been effectively employed in a wide range of applications, e.g., in the estimation of the mean and variance of round trip delay for TCP [27], and in the estimation of jitter for audio playout applications [25, 26]. In all situations the filter weights regulate a tradeoff between the accuracy of the derived estimation (values close to 1) and the ability to quickly detect changes in the input. In the current application the focus is on estimation quality, rather than on responsiveness, so high values should be preferred. The exact identification of filter weights would require some fine tunning which would jointly cater to implementation and operation environment details and would typically remain fixed as has been the case in most similar applications.

If more precise information, regarding the level of network jitter, can be gathered in a central authoritative entity, then it might be meaningful to assign the selection of the appropriate playout policy to that entity, instead of having each receiver decide. A *policy server* could generate/store the playout policies and transmit them to each receiver prior to every video communication.

# 9 Conclusions

This paper has considered the problem of modeling and optimizing a packet video receiver for different levels of delay jitter. To the best of our knowledge, this is the first analytical model capable of capturing a wide area of parameters, including: the level of delay jitter, elaborate intrastream synchronization metrics, and diverse playout policies. The performance evaluation model has been built around the $E_k/D_i/1/N$ queue, which is then generalized into a corresponding Markov decision process that is capable of deriving the optimal playout policy for different levels of delay jitter. The $E_k/D_i/1/N$ queue allows for a sufficient modeling of the actual input traffic (2 moment matching of the input process) and can be solved efficiently by specialized algorithms. The requirement for a traffic model that does not lead to state-space explosition has been a strict one. Notice that the current work is not limited to the performance evaluation of the aforementioned system, but rather proceeds to identify its optimal solution which involves a systematic search in the entire solution space defined by all the possible playout policies and all the possible states. The resulting optimization model has been succesfully used for the derivation of optimal playout policies for a realistic range of delay variability, as identified by actual measurements.

The gain from the optimization under a $k$-Erlang distribution for the modeling of frame inter-arrivals has been demonstrated by a numerical comparison against previous models that make a Poissonian assumption for the input traffic. The presented numerical results have been used to show that: (1) the amount of playout manipulation ought to smooth-out with the regularity of the input traffic; (2) a temporal spreading of discontinuities can be enforced by the playout scheduler. This can potentially lead to their concealment by exploiting human perceptual limitations in the detection of motion. The EO(33) policy derived here utilizes this observation to derive a potentially improved playout quality.

The theoretical optimal playout policy has been transformed into an approximately optimal one that utilizes observable information and it is, thus, feasible to implement. The resulting playout policies may be exploited in implemented systems where the delay jitter is known to vary across different time scales. A dual model of operation, similar in conception to a previous system for spoken voice, can be constructed as follows. A repository of off-line computed playout policies is constructed for different levels of delay jitter, targeting a large scale characterization of traffic conditions (congestion, high load, low load). A receiver uses an estimator to select the policy that best suits the observed conditions and monitors the input traffic, issuing changes of playout policy only under permanent changes in traffic conditions. Transient irregularities in the input (large underflows, clustered arrivals) are handled by the regulation of playout rate as enforced by the derived playout policies.

**Acknowledgements** The authors would like to thank Mr. Nikolaos Labris for helping in conducting the UoA-ASU delay experiment, as well as the anonymous reviewers for their valuable suggestions that helped in improving the overall quality of the paper.

## References


[1] Nikolaos Laoutaris and Ioannis Stavrakakis, "Intrastream synchronization for continuous media streams: A survey of playout schedulers," *IEEE Network Magazine*, vol. 16, no. 3, May 2002.

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

[3] 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.

[4] 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.

[5] Nikolaos Laoutaris and Ioannis Stavrakakis, "An analytical design of optimal playout schedulers for packet video receivers," *Computer Communications*, vol. 26, no. 4, pp. 294–303, Mar. 2003, (An earlier version was presented at the 2nd International Workshop on Quality of Future Internet Services (QofIS2001), Coimbra, Portugal).

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

[7] Eitan Altman, Chadi Barakat, and Victor M. Ramos R., "On the utility of FEC mechanisms for audio applications," in *2nd International Workshop on Quality of Future Internet Services (QofIS2001)*, Coimbra, Portugal, Sept. 2001.

[8] Maria C. Yuang, Shih T. Liang, and Yu G. Chen, "Dynamic video playout smoothing method for multimedia applications," *Multimedia Tools and Applications*, vol. 6, no. 1, pp. 47–60, Jan. 1998.

[9] Marko Luoma, Mika Ilvesmaki, and Markus Peuhkuri, "Source characteristics for traffic classification in differentiated services type of networks," in *Internet III: Quality of Service and Future Directions (VV01), SPIE*, Boston, USA, Sept. 1999.

[10] Cheng Jin and Sugih Jamin, "Design, implementation, and end-to-end evaluation of a measurement-based admission control algorithm for controlled-load service," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July 1998.

[11] B. Adamson, "The mgen toolset," software on-line: http://manimac.itd.nrl.navy.mil/MGEN/.

[12] M. Conti, "Modeling mpeg scalable sources," *Multimedia Tools and Applications*, vol. 13, no. 2, pp. 127–145, Feb. 2001.

[13] A. Chimienti, M. Conti, E. Gregory, M. Lucenteforte, and R. Picco, "Mpeg-2 sources: Exploiting source scalability for an efficient bandwidth allocation," *Multimedia Systems*, vol. 8, no. 3, pp. 240–255, 2000.

[14] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling*, ASA-SIAM series on statistics and applied probability, 1999.

[15] D.M. Lucantoni, "New results on the single server queue with a batch markovian arrival process," *Stochastic Models*, vol. 7, no. 1, pp. 1–46, 1991.

[16] D.M. Lucantoni, K.S. Meier-Hellstern, and M.F. Neuts, "A single server queue with server vacations and a class of non-renewal arrival processes," *Adv. Appl. Prob.*, vol. 22, pp. 676–705, 1990.

[17] Dmitry Loguinov and Hayder Radha, "Large-scale experimental study of internet performance using video traffic," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, Jan. 2002.

[18] A. L. Truslove, "Queue length for the $E_k/G/1$ queue with finite waiting room," *Adv. Appl. Prob.*, vol. 7, pp. 215–226, 1975.

[19] Marcel F. Neuts, *Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach*, Dover Publications, 1995.

[20] G Latouche, P. A. Jacobs, and D. P. Gaver, "Finite markov chain models skip-free in one direction," *Naval Research Logistics Quarterly*, vol. 31, pp. 571–588, 1984.

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

[22] Sheldon M. Ross, *Applied Probability Models with Optimization Applications*, Dover Publications, New York, 1992.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] Henk C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach*, John Wiley & Sons, 1986.

# A Value-iteration algorithm

The value-iteration algorithm is particularly suitable for MDPs with a large state-space. Contrary to policy-iteration and linear-programming solutions – which in each iteration require the solution of a linear system of equations of size equal to the state-space of the problem – the value-iteration algorithm avoids large systems of equations by using a recursive solution from dynamic programming.

The algorithm is based on the computation of a sequence of *value-functions*, $V_n(i)$: $\forall i \in \mathcal{S}$ and $n = 1, 2, \ldots$, which approximate the minimal average cost per unit time. In the following we outline the operation of the algorithm. A good reference for more details is [28].

**Step 0:** Select the initial value function $V_0(i)$ such that: $0 \leq V_0(i) \leq \min_a c_i(a), \forall i \in \mathcal{S}$. Set $n = 1$.

**Step 1:** Update the value function $V_n(i), \forall i \in \mathcal{S}$ by using:

$$V_n(i) = \min_{a \in \mathcal{A}} \left\{ c_i(a) + \sum_{j \in \mathcal{S}} p_{ij}(a) V_{n-1}(j) \right\}$$

and identify the policy $R_n$, which at the $n$th iteration, minimizes the right side of this equation for all $i \in \mathcal{S}$.

**Step 2:** Compute the upper, $M_n$, and lower, $m_n$, bound by using:

$$M_n = \max_{j \in \mathcal{S}} \{V_n(j) - V_{n-1}(j)\} \quad m_n = \min_{j \in \mathcal{S}} \{V_n(j) - V_{n-1}(j)\}$$

The algorithm completes, returning the desired policy $R_{opt} = R_n$, when $0 \leq M_n - m_n \leq \epsilon \cdot m_n$, where $\epsilon$ is a (small) tolerence number. Otherwise, proceed to Step 3.

**Step 3:** Set $n = n + 1$ and go to Step 1.