Playback Delay and Buffering Optimization in Scalable Video Broadcasting

(Invited Paper)

Jean-Paul Wagner, Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - LTS4
CH-1015, Lausanne

Abstract— **This paper addresses the problem of optimizing the playback delay experienced by a population of heterogeneous clients, in video streaming applications. We consider a typical broadcast scenario, where clients subscribe to different portions of a scalable video stream, depending on their capabilities. Clients share common network resources, whose limited rate directly drives the playback delays imposed to the different groups of receivers. We derive an optimization problem, that targets a fair distribution of the playback delays among heterogeneous clients, as well as minimal buffer usage. A server-based scheduling strategy is then proposed, that takes into account the properties of the targeted clients, the channel status, and the structure of the media encoding. A polynomial-time algorithm providing close to optimal results is introduced and it is shown to offer significantly reduced playback delays per client population, as compared to traditional scheduling strategies. In the same time, PSNR performance is not affected, which altogether leads to an overall improvement of the quality of service.**[1]

I. Introduction

Internet video streaming applications usually make use of client buffering capabilities to smooth the discrepancies between the video source rate, and the available channel bandwidth. Buffering then allows for a smooth playback of the stream, but it generally induces a playback delay at the client, and thus impacts the general quality of service.

The particular problem we consider in this paper consists in a broadcast scenario where scalable media is streamed to a variety of heterogeneous clients, such as smart phones, notebooks or workstations. Due to their different capabilities, these clients subscribe to different resolutions of the media stream. They however share a common broadcast channel, whose limited rate directly affects the resolution of the stream that can be sent, and the playback delay induced by buffering at the client. The order in which data from the different hierarchical layers are sent by the server directly influences the distribution of the playback delays among the different receiver groups. The server may decide to first send the lower resolution data, or base layer, and thus to favor the least powerful clients, whose playback delay is then minimal. Such a policy however highly penalizes the other groups of clients, that receive an important share of base layer before the enhancement layers, resulting generally in an increased playback delay.

In this paper, we propose a server-based scheduling strategy that targets a fair distribution of the playback delays among the different groups of clients. It takes into account the network status, the client capabilities, and the video stream characteristics to optimize an average quality of service for all the subscribers. To the best of our knowledge, this work is a first effort to address the playback delay optimization problem, together with and the buffer minimization problem for broadcast to heterogeneous clients.

The paper is organized as follows: we provide an overview of the system under consideration and discuss media scheduling in Section II. In Section III we formalize the considered problem and discuss its implications. Section IV shows our simulation results. Finally we conclude with Section V.

II. Scalable Video Streaming

A. System Overview

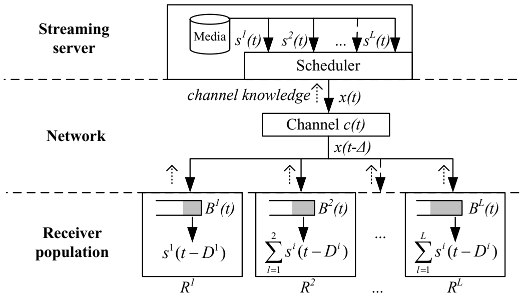

Fig. 1. General overview of the system under consideration.

We consider L-layered hierarchically coded bitstreams that are stored on a streaming server (see Figure 1). In such coding scenarios, all inferior layers from 1 up to $l-1$ must be present at the decoder in order to decode layer l. Depending on the encoding choice, adding a layer may increase the PSNR of the decoded video, the framerate, or the framesize. Each layer is completely determined by its source trace, or playout trace $s^l(t), 1 \leq l \leq L$, indicating how many bits the layer consumes during playout at all time instants t. The channel connecting the server to the receivers is defined by its bitrate

[1] This work has been partly supported by the Swiss National Science Foundation.

0-7803-9319-8/05/$20.00 (c)2005 IEEE.

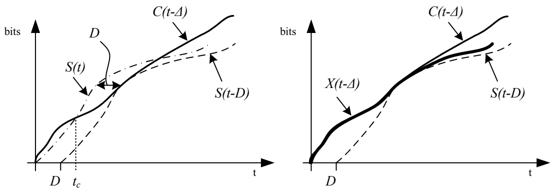

Fig. 2. *Left:* Playback delay and buffer underflow prevention. *Right:* Schedulable play-out trace and a corresponding sending rate trace.

$c(t)$, indicating how many bits the channel is able to transmit at any time t, and a potential network latency Δ. Generally, the server's channel knowledge is extracted from client or network feedback. In this paper we will assume perfect channel knowledge at the server, as offered by guaranteed services for example. For other types of service, our approach leads to an upper bound on achievable performance. L sets of receivers connect simultaneously to the media stream, where each set $R^l, 1 \leq l \leq L$ groups clients that subscribe to all layers up to l of the media stream.

B. Media Scheduling

In such scenarios, the most important logical part of the streaming server is the scheduler: given the source trace and the channel knowledge, it decides when to send data, in order to meet criteria such as desired distortion or delay [1] [2], or maximum utilization of the available channel bitrate. The scheduler outputs a stream of rate $x(t) \leq c(t), \forall t$, the *sending rate*, indicating how many bits are sent on the channel at a given time. After the first bit of the stream is sent by the server, a client in population R^l waits for a time D^l, during which it buffers the data it receives, to ensure that its receiver buffer will never underflow, i.e., the playback will not be disrupted. We call $\mathcal{D} = \{D^l\}_{l=1}^L$ the set of playback delays at the clients.

We will use capital letters (S, C, X) for the cumulative rate functions, e.g., $C(t) = \int_0^t c(u)du$ is the number of bits the channel can transmit up to time t. Note that the cumulative rate functions are all non-decreasing in t. Using this notation [3][4], Figure 2 illustrates de concept of playback delay. If the client starts playback at the reception of the first bit, a buffer underflow occurs at time t_c. Starting playback at the client after D makes sure that the buffer underflow does not occur. We say that a trace $s(t)$ is *schedulable* over a channel with available bandwidth $c(t)^2$, with a playback delay D, if the following condition holds for all t:

$$S(t-D) \quad \leq \quad C(t-\Delta) \tag{1}$$

If condition (1) is met, this implies that the server can find a scheduling such that each of the following, necessary conditions are satisfied for all t:

$$\begin{aligned} S(t-D) &\leq X(t-\Delta) & (2)\\ X(t) &\leq C(t) & (3)\\ x(t) &\leq c(t). & (4) \end{aligned}$$

[2]We assume here that there is no specific flow control policy, and that $c(t)$ is fully usable by the streaming application.

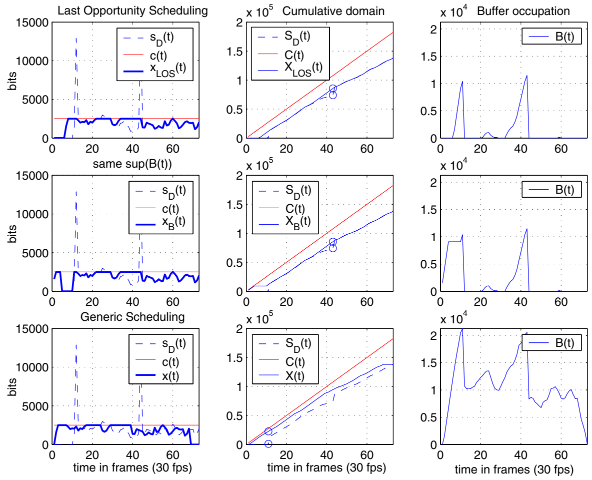

Fig. 3. The video trace is 2 GOPs of the MPEG-4 encoded Foreman sequence. The channel is CBR. The left column depicts the source rate $s(t)$, channel rate $c(t)$ and the illustrated sending rate. The middle column shows the same in cumulative domain, and the right column shows the Buffer occupation as a function of time: $X(t) - S_D(t)$. *Top: LOS* scheduling. The sending rate follows the source rate whenever possible. Whenever $s_D(t) > c(t)$, data is sent at the latest possible earlier opportunity, thus minimizing the buffer occupancy $\forall t$. The maximum amount of buffering needed is 11468 bits (see *top-right*). *Middle:* Another sending rate that minimizes the needed buffer size, but not for all t. $X_B(t) \neq X_{LOS}$. Note that at times 0 to 4 for example bits are put into the buffer that are only retrieved after $D = 10$, and that are only sent later under the *LOS* policy. *Bottom:* A generic sending rate. Note that $\sup_t B(t) = 21264 > 11468$ bits (see *bottom-right*).

In that case, it can be guaranteed that there is a scheduling solution.

C. Last opportunity scheduling LOS

We now introduce the *Last Opportunity Scheduling (LOS)* which will be important in what follows. A *LOS*-Scheduler sends data at the latest possible opportunity given their decoding deadline, so that the overall buffer occupancy is kept small. We set $\Delta = 0$, without loss of generality. If (1) is verified, there exists a family of sending rates $\mathcal{X}$ such that each $X(t) \in \mathcal{X}$ satisfies (2), (3) and (4). There further exists a sending rate in $\mathcal{X}$ that minimizes the buffer occupancy at all times t at the receiver. We will call this sending rate $X_{LOS}(t)$. Suppose that we have $(C(t), S(t), D)$ such that (1) holds. We define $s_D(t)$ as follows:

$$s_D(t) = \begin{cases} 0 & , 0 \leq t < D \\ s(t-D) & , t \geq D \end{cases}$$

$S_D(t)$ is the corresponding cumulative function. $S_D(t)$, as seen from the server, is the number of bits that the receiver has already drained from its buffer at time t, starting at time $t = D$. $X(t)$ is the number of bits that the server has sent to the server up to time t. Thus, at any time t the filling level of the receiver's buffer can be expressed as $X(t) - S_D(t)$. The minimal buffer occupancy is generated by the sending rate:

$$X_{LOS}(t) \quad = \quad \arg\min_{X(t)\in\mathcal{X}} \left(X(t) - S_D(t)\right), \forall t. \tag{5}$$

It is clear that for any sending rate $X(t) \in \mathcal{X}$ s.t. $X(t) \neq X_{LOS}(t)$:

$$X_{LOS}(t) \leq X(t), \forall t \tag{6}$$

Although the *Last Opportunity Scheduler* $LOS\left(c(t), s_D(t)\right)$ is not necessarily the only one generating $X_{LOS}(t)$, it represents a straightforward solution to (5). An illustration is given in Figure 3.

Suppose that we need to transmit a bit that needs to be present at the receiver at its timestamp t_s. As $X(t)$ is a cumulative function, sending the bit earlier than needed, say at time $t_e < t_s$ while having a further opportunity t_{LOS}, $t_e < t_{LOS} \leq t_s$ to send it, increases $X(t)$ for t in $[t_e, t_{LOS}]$, and thus increases also the buffer occupancy at the receiver. It becomes clear that the minimal buffer occupancy is given by the sending rate $X_{LOS}(t)$, as defined in (5).

LOS Scheduling will be important in what follows to construct a set of playback delays for a set of receivers $\{R^l\}$. Indeed, it can also be shown that for a fixed set of playback delays $\mathcal{D}$, using *LOS* on each layer of the stream individually, the buffer occupancy at each set of receivers R^l is minimized at all times, given $\mathcal{D}$.

III. Playback Delay optimization

A. Problem Formulation

Consider a channel given by its cumulative rate trace $C(t)$, and a set of L hierarchically coded layers given by their cumulative source rate traces $\{S^l\}_{l=1}^{L}$. The channel connects a streaming server to L sets of receivers $\{R^l\}_{l=1}^{L}$, that simultaneously subscribe to layers up to l. Our aim is to find a set of playback delays $\mathcal{D} = \{D^l\}_{l=1}^{L}$, $D^1 \leq D^2 \leq \ldots \leq D^L \leq D_{max}$, that minimizes a global metric $\varphi(\cdot)$ over the set of possible playback delay sets:

$$\mathcal{D}_f \quad = \quad \arg\min_{\mathcal{D}} \left(\varphi(D^0, ..., D^L)\right) \tag{7}$$

such that for any $l \leq L$, $S^l_{\mathcal{D}}(t) \leq C(t)$, where $S^l_{\mathcal{D}}(t) = \sum_{i=1}^{l} S^i\left(t - D^i\right)$. This is, from (1), a sufficient condition for the trace $S^l_{\mathcal{D}}(t)$ to be schedulable over the channel $c(t)$. Let D^l_{min} denote the smallest possible playback delay for layers up to l. In order to have a *fair* distribution of the penalty on the playback delays, we choose to minimize the standard deviation of the relative penalties induced by a set of playback delays $\mathcal{D}$. The global metric $\varphi(\cdot)$ becomes:

$$\varphi(D^0, ..., D^L) = \sigma\left((D^0 - D^0_{min}), ..., (D^L - D^L_{min})\right). \tag{8}$$

In the same time, we want to minimize the buffer occupancy at the receivers for a given set of playback delays. We propose to solve the optimization of the playback delays, and buffer occupancy in two steps, using the previously outlined *LOS* strategy.

B. Playback Delay Analysis

In this subsection we will introduce some general results, inspired from [4], that are important in order to formally derive lower bounds on the playback delays, which is a crucial step towards solving (7).

Suppose that we have two increasing non-zero functions $F(t)$ and $G(t)$ such that $\lim_{t\to\infty} F(t) \geq \lim_{t\to\infty} G(t)$. We define the (maximum) horizontal distance between $F(t)$ and $G(t)$ as follows:

$$h(G, F) \quad = \quad \sup_t \left(F^{-1}\left(G(t)\right) - t\right), \tag{9}$$

where $F^{-1}(t) = \min\{t : F(t) \geq x\}$ is a pseudo-inverse of $F(t)$. The following relations hold:

$$\begin{aligned}
h(G,F) = 0 \quad &\Leftrightarrow \quad F(t) \geq G(t), \forall t \text{ and} && (10)\\
& \qquad\quad \exists\tau \text{ s.t. } F(\tau) = G(\tau) && (11)\\
h(G,F) < 0 \quad &\Leftrightarrow \quad F(t) > G(t), \forall t && (12)\\
h(G,F) > 0 \quad &\Leftrightarrow \quad \exists\tau \text{ s.t. } F(\tau) < G(\tau). && (13)
\end{aligned}$$

Two useful properties of $h(\cdot)$ will be used in the optimization :

1) If $h(G, F) > 0$ and $G'(t) = G(t - h(G, F))$, then $h(G', F) = 0$. In other words, $h(G, F)$ is the minimum shift we need to apply on $G(t)$, so that $F(t) \geq G'(t)$, $\forall t$.
2) Let $F(t)$, $G(t)$ and $G'(t)$ be non-decreasing functions such that $G'(t) > G(t)$, $\forall t$. Then: $h(G', F) > h(G, F)$. Indeed by the definition of $h(\cdot)$ and $F^{-1}(\cdot)$, and because $F(t)$ is non-decreasing, the result follows immediately, as $F^{-1}\left(G'(t)\right) > F^{-1}\left(G(t)\right)$, $\forall t$. Similarly, if $G'(t) < G(t)$, $\forall t$ then $h(G', F) < h(G, F)$.

Let $\vec{\delta}$ denote any set of L decreasingly ordered positive values: $\vec{\delta} = \{\delta_1, \delta_2, \ldots, \delta_L\}$, $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_L \geq 0$. We will use the following notation $\forall l$, $1 \geq l \geq L$: $G^l_{\vec{\delta}}(t) = \sum_{i=1}^{l} G^i\left(t + \delta_i\right)$ and $G^l_0(t) = \sum_{i=1}^{l} G^i(t)$.

Lemma 3.1: Consider a set of L non-decreasing functions $\{G^l(t)\}_{l=1}^{L}$ and a non-decreasing function $F(t)$, all defined on the temporal axis. We have, $\forall l$, $1 \geq l \geq L$ and $\forall\vec{\delta}$:

$$D^l_0 = h\left(G^l_0, F\right) \quad < \quad h\left(G^l_{\vec{\delta}}, F\right) = D^l_{\vec{\delta}} \tag{14}$$

Proof: As the functions $\{G^l(t)\}_{l=1}^{L}$ are non-decreasing, we have, $\forall l$, $1 \geq l \geq L$ and $\forall\delta_l > 0$: $G^l(t) < G^l\left(t + \delta_l\right)$. Thus, $\forall l$, $1 \geq l \geq L$ and $\forall\vec{\delta}$:

$$G^l_0(t) \quad < \quad G^l_{\vec{\delta}}(t) \ ,\forall t \tag{15}$$

From above, it follows that $D^l_0 < D^l_{\vec{\delta}}$. ■

C. Minimal delay for one set of receivers

Applying this property to cumulative source rate traces, we derive a lower bound on the playback delay for the clients in set R^l. If $S^l_0(t) > C(t)$, for some t, the smallest playback delay for layer l, is given by:

$$D^l_0 \quad = \quad h\left(S^l_0, C\right), \tag{16}$$

where $S^l_0(t) = \sum_{i=1}^{l} S^i(t)$ is the sum of all the layers without relative shifts. Since layers are hierarchically ordered, we know, by application of Lemma 3.1, that D^l_0 is the lower bound on all possible playback delays for layer l, thus $D^l_{min} = D^l_0$. Furthermore, as all the rate traces are positive valued functions, $S^{l+1}_0(t) \geq S^l_0(t)$, $\forall t$, so from (III-B) we have $D^{l+1}_{min} \geq D^l_{min}$.

Note that any playback delay lower than D^l_{min} will result in a buffer underflow at the receiver, while any larger playback delay will allow reception without experiencing a buffer underflow. This allows us to formulate a quick algorithm to find D^l_{min} using a simple bisection search, see Algorithm 1. It is important to note that, by achieving the minimum playback delay for a given layer l, we do not necessarily achieve the minimum playback delay for any other layer. It rather represents a bound that will be used in the optimization algorithm that aims at solving (7). This can be illustrated using a simple 2-layer example, depicted in Figure 4. In Figure 4-top, we set $D^1 = D^1_{min}$, without considering higher layers. In that case playout of layer 1 can begin after $D^1_{min} = 2$ frames. If we consider layer 2 (Figure 4-middle) without taking into account lower layers individually, playout of layers 1 and 2 can begin after $D^2_{min} = 127$ frames. Note the induced playback delay penalty of 125 frames for clients in set R^1. Figure 4-bottom shows the *greedy* rate allocation scheme: D_1 is fixed to D^1_{min} and the playback delay for layer 2 is computed using the remaining channel bitrate. The playback delay for layer 2 grows to 219 frames, inducing a relative penalty of 92 frames. We are finally facing a typical tradeoff situation: as we increase the relative playback delay penalty for lower layers, we leave more available channel bits that can be used to decrease the playback delay penalty for higher layers.

Algorithm 1 $D_{min} = getDmin\left(C(t), S(t)\right)$

```
$D_{low} \Leftarrow 0$
$D_{high} \Leftarrow$some large value
while $\left(D_{high} - D_{low}\right) > 1$ do
    $D_{test} \Leftarrow \left\lfloor \frac{D_{low}+D_{high}}{2} \right\rfloor$
    if $S\left(t - D_{test}\right) \leq C(t), \forall t$ then
        $D_{high} \Leftarrow D_{test}$
    else
        $D_{low} \Leftarrow D_{test}$
    end if
end while
$D_{min} = D_{test}$
```

D. Optimization Algorithm

Finding $\mathcal{D}_f = \{D^l_f\}^L_{l=1}$, the solution to the global optimization problem, is the hardest part of the joint problem. Once we have $\mathcal{D}_f$, we know how to schedule the layers in order to minimize the buffer occupancy, namely we need to use *LOS* scheduling. Finding $\mathcal{D}_f$ is a combinatorial problem, and solving it generally implies a full search algorithm which belongs to class *NP*. Based on the example introduced above we can however make some observations about the structure of the solution space. We will first consider *greedy* layered scheduling as given by Algorithm 2. Using the greedy allocation scheme, the playback delay is minimized for the first layer given the available bitrate, and using the remaining bitrate, we iterate on higher layers. This scenario results in an upper bound D^L_g on the playback delay for the highest layer L, in the sense that increasing D^L beyond D^L_g will no longer liberate channel bits that could be used to minimize the playback delays of lower layers. Indeed, fixing each $D^l_g, l < L$ to the shortest possible delay, given the available bitrate at that iteration, which results from the same procedure on the previous layer, all the spare bitrate for layer L will be available at the latest possible instant in time. We also have seen that we can easily compute D^L_{min}. We can thus drastically limit the range of values in which D^L can evolve. Starting from there, we can loop through the possible values of D^L in the range $[D^L_{min}, D^L_g]$. Fixing a value D^L, we compute the bitrate available for layers 1 to $L-1$ as: $C^{L-1}(t) = C(t) - X^L(t)$. Given this channel bitrate, we then compute the range of possible delays for layer $L-1$, $[D^{L-1}_{min}; D^L]$, fix a value D^{L-1} and iterate down through the lower layers. While searching, we are thus limiting the search space to only feasible solutions. Although this reduces the number of iterations of the search algorithm, the worst case complexity remains unchanged.

Algorithm 2 $\left(\left\{x^l_g\right\}, \left\{D^l_g\right\}\right) = Greedy\left(c(t), \left\{s^l(t)\right\}\right)$

```
$c^1(t) \Leftarrow c(t)$
for $l = 1$ to $L$ do
    $D \Leftarrow getDmin\left(C^l(t), S^l(t)\right)$
    $x^i_g = LOS\left(c^l(t), s^l_D(t)\right))$
    for all $t$ such that $0 \leq t \leq T + D$ do
        $c^{l+1}(t) \Leftarrow c^l(t) - x^l_g(t)$
    end for
    $D^l_g = D$
end for
```

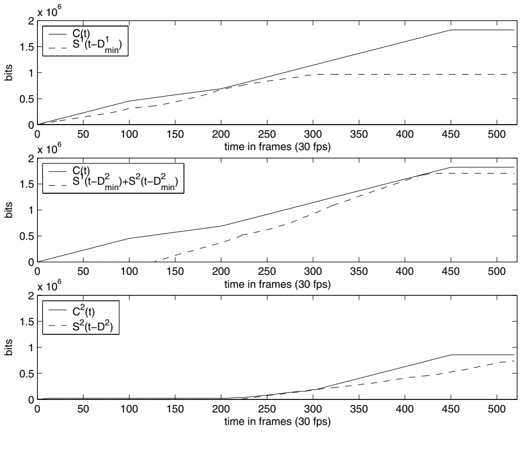

Fig. 4. Scheduling 300 frames of Foreman (QCIF). The 2 layers represent the MPEG-4 FGS base layer and the first bitplane of the enhancement layer. The channel is available for 450 time units, providing a mean rate of 128kbps, which drops to 64kbps between times 100 and 200.

E. Heuristic-based Algorithm

We therefore consider a sub-optimal quick heuristic to find an approximation of $\mathcal{D}_f$. It is based on the a priori information we have about the structure of the optimal solution. Indeed, any set of delays $\mathcal{D}_h = \left\{D^l_h | D^l_h = D^l_{min} + k_l\right\}^L_{l=1}$, where k_l are equal positive integers ($k^l = K, 1 \leq l \leq L$) sets $\varphi\left(\mathcal{D}_h\right) = 0$. In other words the source traces of all layers need to be delayed by K units relative to their respective minimal playback delay D^l_{min}. Given the set of minimum playback delays $\mathcal{D}_{min}$, we thus construct the aggregate source rate trace $S^L_{\mathcal{D}_{min}}(t)$, defined as:

$$S^L_{\mathcal{D}_{min}}(t) \quad = \quad \sum_{i=1}^{L} S^i\left(t - D^i_{min}\right) \qquad (17)$$

Using this, we can compute K as the minimum delay that is needed to receive $S^L_{\mathcal{D}_{min}}(t-K)$ over the channel of rate $C(t)$. Thus:

$$K \quad = h\left(S^L_{\mathcal{D}_{min}}, C\right) \qquad (18)$$

So $k_l = K, \forall l$ can be computed by using Algorithm 1 once.

We know that an upper bound on the playback delay for the highest layer L is given by D^L_g, so we can reduce the value we found to $D^L_h = min\left(D^L_h, D^L_g\right)$, by adjusting k_L accordingly. We then reduce the value for layer $L-1$ in the same way: the upper bound is derived by greedily scheduling the $L-1$ lower layers over the channel of bitrate $C^{L-1}(t) = C(t) - X^L(t)$. Iterating this reduction procedure on all the lower layers gives us the final result. This algorithm executes in polynomial time and our simulations show that it finds results close to the optimum. If the optimum set of playback delays is such that $\varphi\left(\mathcal{D}_f\right) = 0$, the algorithm finds the optimum itself.

IV. Results

Foreman		
	D^2_{min}	$\mathcal{D}_f$
D^1	212	54
D^2	212	259

Composite		
	D^2_{min}	$\mathcal{D}_f$
D^1	241	59
D^2	241	297

TABLE I

Fair playback delays $\mathcal{D}_f$ in frame units. Channel: 100kbps CBR.

The video traces we used in our simulations are Foreman (300 frames), and a composite sequence made up of Foreman, Coastguard and News (total of 900 frames), both QCIF at 30 Hz framerate. We used the MoMuSys MPEG-4 FGS [5] reference codec to encode the sequences into a base layer and an enhancement layer. The *GOP* size is 150 frames, and it only contains P-frames. The enhancement layer has been cut along the bitplane boundaries to construct further layers.

Table I shows results for both sequences sent over a channel of mean rate 100kbps. The channel can transmit 2 layers in both cases. The gain in playback delay for receivers of set R^1 is of the order of seconds when using our fair distribution.

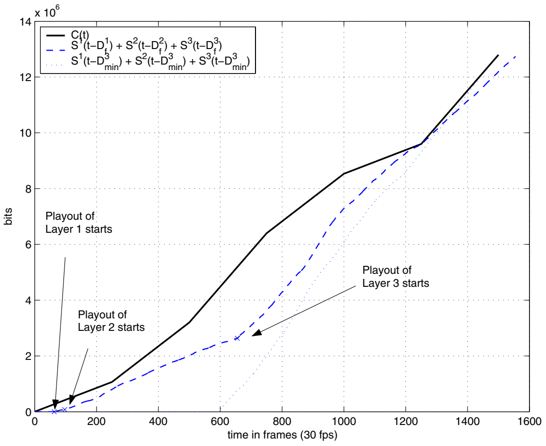

Fig. 5. Dashed curve: aggregate playout curve with playback delays $\mathcal{D}_f$. The aggregate playout curve using playback delay D^3_{min} is shown for reference.

	D^3_{min} (reference)	$\mathcal{D}_f$ (optimum)	$\mathcal{D}_h$ (heuristic)
D^1	594	63	63
D^2	594	97	97
D^3	594	655	655
iterations	n/a	4824189	10

TABLE II

Optimal VS heuristic performance (delays in frame units)

Figure 5 and Table II show the results of another simulation run: we consider sending the composite sequence over a piecewise CBR channel with rate changing between 128, 256 and 384kbps, which can transmit 3 layers. Using a fair playback delay distribution playout at receivers of sets R^l, $l = 1, 2, 3$ can start playback after a delay of D^l, $l = 1, 2, 3$ respectively, as given in Table II. Note the gain in delay for clients in sets R^1 and R^2, compared to a playback delay of 594 frames if D^3_{min} is used for all clients (dotted line in Figure 5). The relative playback delay penalty per client set, compared to their respective D^l_{min} value, is of 61 frames each. The *iterations* field in Table II gives the number of feasible combinations that have to be checked for optimality for $\mathcal{D}_f$, which grows exponentially with the number of layers. The checking itself is performed in polynomial time. Similarly, *iterations* for $\mathcal{D}_h$ indicates the number of bisection search iterations before the result was found. This number grows logarithmically with the length of the trace only. Each iteration only contains steps of polynomial complexity.

V. Conclusions

In this paper, we have outlined and formalized the problem of playback delay distribution in a scalable streaming scenario, where a streaming server broadcasts to a heterogeneous set of clients. We have proposed a server-based scheduling strategy, and validated a computationally fast algorithm, that targets a fair distribution of the playback delays and minimizes the buffering at the receivers. It is shown to bring significant improvements on the playback delays experienced by the clients, since it takes into account the heterogeneities in the client population, the structure of the encoded stream, and the available channel knowledge.

References

[1] P.A. Chou, Z. Miao, *Rate-distortion optimized streaming of packetized media*, Microsoft Research Technical Report MSR-TR-2001-35, February 2001.

[2] P. De Cuetos, K.W. Ross *Unified Framework for Optimal Video Streaming*, Infocom 2004, Hong Kong, February 2004.

[3] T. Stockhammer, H. Jenkac, G. Kuhn, *Streaming Video over Variable Bit-Rate Wireless Channels*, IEEE Transactions on Multimedia, Vol. 6, No.2, April 2004.

[4] P. Thiran, J.-Y. Le Boudec, *Network Calculus*, Springer-Verlag Lecture Notes on Computer Science number 2050.

[5] W. Li, *Overview of Fine Granularity Scalability in MPEG-4 Video Standard*, IEEE Transactions on Circuits and Systems for Video Technology, March 2001.