Playout Adaptation for Peer-to-peer Streaming Systems under Churn

Ilias Chatzidrossos and György Dán
School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden
Email: {iliasc, gyuri}@kth.se

***Abstract*—We address the problem of playout adaptation in peer-to-peer streaming systems. We propose two algorithms for playout adaptation: one coordinated and one distributed. The algorithms dynamically adapt the playback delay of the peers so that the playout miss ratio is maintained within a predefined interval. We validate the algorithms and evaluate their performance through simulations under various churn models. We show that playout adaptation is essential in peer-to-peer systems when the system size changes. At the same time, our results show that distributed adaptation performs well only if the peers in the overlay have similar playback delays. Thus, some form of coordination among the peers is necessary for distributed playout adaptation in peer-to-peer streaming systems.**

I. Introduction

Playout adaptation is widely used in multimedia communication to provide continuous playback of audio and/or video information despite network induced impairments, such as delay jitter and fast varying network throughput. To make playout adaptation possible, the receiver stores the received data in a playout buffer before eventually playing it out. Playout adaptation consists then of two tasks. At the beginning of a connection the receiver has to decide when to start the playout and potentially what to start the playout with; later on it aims to avoid that the playout buffer becomes empty or that it overflows by, for example, adapting the playback rate [1] or by rescheduling the playback of the subsequent talksuprts during silence periods [2].

In peer-to-peer (P2P) streaming systems, the audio or video data sent by a server can be forwarded by several peers before it reaches a particular peer. While data forwarding saves server upload bandwidth and cost, it makes the data delivery process subject to several factors that are hard to control or to predict. On one hand, the forwarding algorithms used by the peers together with the varying upload bandwidths of the peers result in a complex multi-path data distribution process, in which out-of-order data delivery is not the exception but the rule [3]. On the other hand, the peer arrivals and departures disturb the data forwarding and also influence the time needed to distribute the data through changing the number of peers in the system [4].

Thus, playout adaptation in P2P systems should not lead to unnecessary adaptation, despite the highly stochastic nature of the data distribution process. At the same time, it must respond promptly to changes in the chunk missing ratio of the peers, caused by, for example a sudden increase in the number of peers. To meet these challenges, playout adaptation in P2P streaming systems can potentially make use of more information than in the case of point-to-point communication. To support playout adaptation, the peers in the system might use information from their neighbors, or could rely on global information, obtained from a central or from a distributed entity. Nevertheless, the playout adaptation performed by individual peers might affect the data delivery to other peers, thus the problem is inherently complex.

In this paper we address the problem of playout adaptation in P2P live streaming systems. We propose a centralized and a distributed algorithm to adapt the playback delay such as to maintain the system in a desired operating regime. We use extensive simulations to show that the algorithms are beneficial in a steady state system, and they provide significant gains when the number of peers changes.

The rest of the paper is organized as follows. In Section II we review related work. In Section III we describe the considered P2P streaming system and formulate the playout scheduling and adaptation problem and in Section IV we present the two adaptation algorithms. Section V provides a simulation-based evaluation of the algorithms. Section VI concludes the paper.

II. Related work

Playout scheduling and adaptation has received significant attention for the case of point-to-point media transmission. For streaming media, playout schedulers rely on time or on buffer occupancy information [5]. In time-based playout schedulers, timing information is embedded in the packets so that the receiving host can measure absolute or relative differences in the packet delivery times [6]. In buffer-based playout schedulers the network's condition is inferred from the evolution of the buffer occupancy [7].

Playout adaptation can be performed in various ways, depending on the type of media. In the case of voice communications, schedulers can leverage the silence periods and postpone the playout of the next talkspurt to ensure playout continuity [2]. In the case of video/audio streaming, playout adaptation can be performed by either frame (packet) dropping or by altering the frames' playout duration. In the case of frame dropping, the receiver drops frames from the playout buffer to avoid buffer overflow or freezes the playout to avoid buffer underflow [8], [9]. By altering the frame duration the

receiver can decrease the rate of the playout so that the buffer builds up and can decrease the playback delay if the playout buffer occupancy is high [1], [7]. In [1], a window-based playout smoothing algorithm changes the playout rate at the receiver based on the current buffer occupancy and on the traffic during the next time window as predicted by a neural network. In terms of objectives our work is more similar to [7], where an adaptive media playout algorithm selects the playout rate according to the channel conditions in order to achieve low latency for a given buffer underflow probability.

Contrary to point-to-point communication, playout adaptation for P2P streaming has not received much attention. In [9], the authors compare delay-preserving and data-preserving playout schedulers and conclude that a playout scheduler should keep the peers synchronized to improve the data exchange efficiency. They consider however, tree-based systems where there are only peer arrivals. In [10], a playout adaptation scheme for tree-based systems is proposed that allows peers to adjust their playout rate in order to reduce the average playback delay in the system. Two processes performed locally at the peers, catching-up and parent-reversal, are defined so that the propagation delay in the resulting overlay trees is minimized. Nevertheless, none of these works consider the need for playout adaptation due to the changes of the overlay size. To the best of our knowledge, our work is the first that addresses the playout scheduling and adaptation problem for P2P streaming and focuses on mesh-based streaming systems, which constitute the vast majority of deployed P2P streaming systems nowadays.

III. Problem Formulation

We consider a P2P live streaming system consisting of a streaming server and a set $\mathcal{N}(t)$ of peers. Peers arrive to the system according to a counting process $A(t)$, and depart with intensity $\mu(t)$. We denote the number of peers at time t by $N(t)$, the arrival time of peer n by T_A^n and its departure time by T_D^n. The streaming server and the peers maintain an overlay, and use the overlay connections to distribute the video stream generated by the streaming server to the peers. The video stream consists of a sequence of chunks of data, e.g., data corresponding to a few frames or a group of frames, generated at regular time intervals δ, and we denote the time when chunk c is generated at the streaming server by t_c. To distribute the stream, the peers relay the chunks among each other according to some forwarding algorithm, e.g., [11], [12]. In order to make relaying possible, each peer stores the chunks that it has received in a buffer until the chunks are played out.

We denote the time when chunk c is received by peer n by t_c^n. The time $t_c^n - t_c$ it takes for chunk c to be delivered to peer n depends on many factors, such as the number of peers, the overlay structure, the peers' upload rates, the fowarding algorithm, and is in general hard to predict. Consider now an arbitrary peer n and the sequence of chunk arrival times $t_c^n, t_{c+1}^n \ldots$. Peer n can only play out chunk c after it receives it, i.e., at some time $t \geq t_c^n$. We refer to the difference $t - t_c = B^n(t_c^n)$ as the playback delay used for chunk c. We refer to the share of chunks that a peer cannot play out on time as the chunk missing ratio. For peer n and a time interval $\iota = [T_1, T_2]$ we define the chunk missing ratio as

$$\pi^n(\iota) = \frac{|\{c : t_c^n \in \iota \cap [T_A^n, T_D^n] \wedge B^n(t_c^n) < t_c^n - t_c\}|}{|\{c : t_c^n \in \iota \cap [T_A^n, T_D^n]\}|} \quad (1)$$

Furthermore, we define the average chunk missing ratio $\pi(\iota)$ calculated over all peers in the system in the interval ι.

The playback delay $B^n(t)$ has to be chosen upon arrival, but need not be constant until the departure of the peer. As shown by recent work [13], the perceived visual quality is rather insensitive to minor variations of the frame rate. According to the model of perceived visual quality proposed in [13], the mean opinion score (MOS) can be expressed as a function of the frame rate f as

$$MOS(f) = Q_{max} \frac{1 - e^{-\gamma \frac{f}{f_m}}}{1 - e^{-\gamma}}, \quad (2)$$

where γ and Q_{max} are video specific constants and f_m is the maximum frame rate.

Consider now that the frame rate is changed to $f_a = (1+a)f$ for some small $a \approx 0$. For $a \approx 0$ the derivative of the MOS score can be approximated by

$$\frac{dMOS(f_a)}{da} \approx Q_{max} \gamma \frac{f}{f_m} \frac{e^{-\gamma \frac{f}{f_m}}}{1 - e^{-\gamma}}, \quad (3)$$

by using that $e^a|_{a=0} = 1$. Observe that if (2) is high then (3) is low. Similarly, it has been observed that the perceived quality is not very sensitive to small variations of the playback rate [1], [7]. We thus consider that the playback delay $B^n(t)$ of a peer can be adapted by accelerating or decelerating the playback. We denote the maximum rate of adaptation by $\hat{a} > 0$, and assume that the effect of adaptation on the perceived visual quality is negligible at adaptation rates $|a| < \hat{a}$, similar to the assumption in [1], [7].

For peer n to be able to play out all chunks, it has to adapt its playback delay $B^n(t_c^n)$ such that $t_c + B^n(t_c^n) \geq t_c^n$ for all chunks c that it should play out in the time interval $[T_A^n, T_D^n]$. If all peers adapt their playback delay this way then $\pi = 0$, but this might be difficult to achieve in practice due to the dynamics of the system. Thus, we consider that a chunk missing ratio no higher than $\tau > 0$ is acceptable, and can be compensated for by some form of channel coding. At the same time if channel coding capable of compensating for a chunk missing ratio of τ is used, then the actual chunk missing ratio should not be below $\eta\tau$, where $1 > \eta > 0$, because then the bandwidth used for channel coding would be wasted. Thus, we consider that the goal of playout adaptation is to adjust the playback delay $B^n(t)$ of the peers such that the chunk missing ratio stays within the band $[\eta\tau, \tau]$, subject to the constraint on the maximum rate of adaptation $\hat{a}$.

We measure the performance of playout adaptation over a time interval $\iota = [T_1, T_2]$ by the square error of the chunk missing ratio calculated over the interval ι,

$$SE(\iota) = ([\eta\tau - \pi(\iota)]^+)^2 + ([\pi(\iota) - \tau]^+)^2 \quad (4)$$

Algorithm 1 Adaptive band control algorithm for CA

1: Input: $\pi(\iota)$ for $\iota = [T_i - B(T_i), T_i]$
2: $\overline{\sigma} = (1-\beta)\cdot\overline{\sigma} + \beta\cdot|\overline{\pi} - \pi(\iota)|$, $(\beta = 1/4)$
3: $\overline{\pi} = (1-\alpha)\cdot\overline{\pi} + \alpha\cdot\pi(\iota)$, $(\alpha = 1/8)$
4: **if** $\pi(\iota) > \tau$ **then**
5: **if** $\overline{\pi} > \tau\ OR\ \pi(\iota) > \overline{\pi} + \kappa\cdot\overline{\sigma}$ **then**
6: **if** $previous\_action == INCREASE$ **then**
7: $\gamma = 2\cdot\gamma$
8: **else**
9: $\gamma = max(\gamma - 1, 1)$
10: **end if**
11: $B' = B + \gamma\cdot\delta$
12: $previous\_action \leftarrow INCREASE$
13: **end if**
14: **else if** $\pi(\iota) < \eta\cdot\tau$ **then**
15: **if** $\overline{\pi} < \eta\cdot\tau\ OR\ \pi(\iota) < \overline{\pi} - \kappa\cdot\overline{\sigma}$ **then**
16: $\gamma = max(\gamma - 1, 1)$
17: $B' = B - \delta$
18: $previous\_action \leftarrow DECREASE$
19: **end if**
20: **else**
21: $\gamma = max(\gamma - 1, 1)$
22: **end if**

where $[.]^+$ denotes the positive part. When the SE is averaged over consecutive intervals ι, then we get the mean square error (MSE) of the chunk missing ratio. The MSE quantifies the average deviation of the chunk missing ratio from the target interval.

IV. Playout Adaptation Algorithms

In the following we present two playout adaptation algorithms. The first one, called *Coordinated Adaptation (CA)*, is a centralized algorithm and lets all peers have the same playback delay calculated based on the average chunk missing ratio. We use this algorithm mainly to illustrate the importance of playout adaptation in P2P streaming. The second one, called *Distributed Adaptation (DA)*, allows each peer to adapt its playback delay independently of the other peers, based on the chunk missing ratio it experiences.

A. Coordinated Adaptation (CA)

In the case of coordinated adaptation the playback delay is recalculated periodically by executing the adaptive band control algorithm shown in Algorithm 1, and is the same for all peers in the system.

At the i^{th} execution of the algorithm, at time T_i, the input to the control algorithm is the average chunk missing ratio $\pi(\iota)$ over the interval $\iota = [T_i - B(T_i), T_i]$. This is used to calculate the exponentially weighted moving average $\overline{\pi}$ of the chunk missing ratio and its deviation $\overline{\sigma}$ (lines 2-3 in Algorithm 1), similar to the round-trip time estimation in TCP [14], using $\alpha = 0.125$ and $\beta = 0.25$. The algorithm uses $\overline{\pi}$, $\overline{\sigma}$, and the most recent average chunk missing ratio $\pi(\iota)$, to decide how to change the playback delay. The playback delay B is increased if both the average chunk missing ratio and its moving average exceed the target τ or if the average chunk missing ratio exceeds the target τ and is higher than the moving average plus a constant $\kappa > 0$ times its deviation (lines 6-12). The second condition resembles the RTO calculation used in TCP [14]. By tuning κ we can control the trade-off between fast adaptation to changing conditions and unnecessary adaptation in a stationary system. The conditions for decreasing the playback delay are similar (lines 15-18).

Adaptation is performed in time units equal to the chunk time δ. The amount of time units γ by which the playback delay is increased or decreased is determined by the history of adaptations performed by the algorithm. Initially, $\gamma = 1$. This value is doubled every time the playback delay has to be increased consecutively, and is decremented by one otherwise. The adaptation of γ resembles the additive increase multiplicative decrease used in TCP congestion control [14], and allows for a fast increase of the playback delay if needed, but leads to a smaller rate of decrease.

If the playback delay is changed, the peers are informed about the new playback delay B', and accelerate or decelerate their playout rate to adapt their playback delay to B'. The time needed to perform the adaptation is $|B' - B|/\hat{a}$. The next time the control algorithm is executed is time B' after the adaptation has finished, that is, at time $T_{i+1} = T_i + |B' - B|/\hat{a} + B'$. The arriving peers set their playback delay to $B^n(T_A^n) = B'$ upon arrival and then adapt their playback delay as dictated by the *CA* algorithm.

B. Distributed Adaptation (DA)

In the case of distributed adaptation the playback delay is recalculated periodically by every peer through executing the band control algorithm shown in Algorithm 2. The input to the control algorithm is the length s_j of the last eight loss intervals (i.e., the number of chunks received between missed chunks), which is used to estimate the average chunk missing ratio using the Average Loss Intervals method as in TFRC [15].

The algorithm uses the estimated chunk missing ratio to decide how to change the playback delay. The playback delay is increased if the average chunk missing ratio exceeds the target τ, is decreased if the average chunk missing ratio is lower than $\eta\tau$, and is left unchanged otherwise. The adaptation step size γ is adjusted the same way as in the case of Algorithm 1, i.e., it is doubled upon consecutive increases of the playback delay, and is decremented by one unit otherwise. Once the new playback delay B' is calculated, the peer accelerates or decelerates its playout rate to adapt its playback delay to the new value. The time needed for adaptation is $|B' - B|/\hat{a}$. The next time the control algorithm is executed is time B' after the adaptation has finished, similar to *CA*.

We consider three policies that the peers can use to choose their initial playback delay $B^n(T_A^n)$ upon arrival to the system.

- Global: The initial playback delay is the average playback delay of the peers in the overlay, i.e., $B^n(T_A^n) = \frac{1}{|\mathcal{N}(T_A^n)|}\sum_{j\in\mathcal{N}(T_A^n)} B^j(T_A^n)$

Algorithm 2 Band control algorithm for DA

$w \leftarrow (1, 1, 1, 1, 0.8, 0.6, 0.4, 0.2)$
$\hat{s} \leftarrow \frac{\sum_{j=1}^{8} w_j \cdot s_j}{\sum_{j=1}^{8} w_j}$
$\pi = \frac{1}{\hat{s}+1}$
if $\pi > \tau$ **then**
  if $previous\_action == INCREASE$ **then**
    $\gamma = 2 \cdot \gamma$
  else
    $\gamma = max(\gamma - 1, 1)$
  end if
  $B' = B + \gamma \cdot \delta$
  $previous\_action \leftarrow INCREASE$
else if $\pi < \eta \cdot \tau$ **then**
  $\gamma = max(\gamma - 1, 1)$
  $B' = B - \delta$
  $previous\_action \leftarrow DECREASE$
else
  $\gamma = max(\gamma - 1, 1)$
end if

- Local: The initial playback delay is the average playback delay of the neighboring peers, i.e., $B^n(T_A^n) = \frac{1}{|\mathcal{D}_n(T_A^n)|} \sum_{j \in \mathcal{D}_n(T_A^n)} B^j(T_A^n)$, where $\mathcal{D}_n(T_A^n)$ is the set of neighbors of peer n.
- Fixed: The initial playback delay is $B^n(T_A^n) = B_0 > 0$.

Of these three policies, the Local policy is easiest to implement in a distributed way: An arriving peer can schedule for playback the chunk that its average neighbor would play out. The other two policies would require a globally synchronized clock, and therefore we use them mainly as a basis for comparison.

V. Performance evaluation

In the following we use simulations to evaluate the proposed algorithms and to get an insight into their operation.

A. Simulation Methodology

We implemented the playback adaptation algorithms in the publicly available packet-level event-driven simulator P2PTV-SIM [16]. We implemented an overlay management algorithm that maintains an overlay graph in which every peer has between $0.5d$ and d neighbors: arriving peers try to establish d connections to randomly selected peers, potentially by letting some peers drop one of their already established connections. If the number of neighbors of a peer drops below $0.5d$ then it tries to connect to new neighbors. In our simulations, we use $d = 20$.

The video stream has a rate of 1 Mbps and is segmented into equally sized chunks of 100 kbits, having thus $\delta = 0.1$ s. Chunk forwarding between the peers is based on the pull-token algorithm from [12]. The download bandwidth of the peers is unlimited, while for the upload bandwidth, we use the distribution shown in Table I. In order to capture the effect of network delays, peers are dispersed over seven geographical regions according to the network model presented in [17] with an average latency between a pair of peers of 96 ms. For the adaptation algorithms *CA* and *DA*, we use $\hat{a} = 0.05$ and a value of $\kappa = 2$, which we found to yield a good trade-off between fast response to the chunk missing ratio and unnecessary adaptation of the playback delay.

Class	Upload (Mbps)	Peers (%)
1	5	10
2	1.5	10
3	1	40
4	0.55	40

TABLE I
Peer Bandwidth Classes

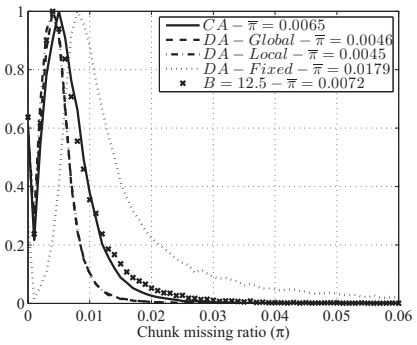

Fig. 1. Normalized histogram of the chunk missing ratio for the *CA* algorithm and the *DA* algorithm with different startup policies. The target miss ratio is $\tau = 0.01$ and $\eta = 0.5$. Overlay of $\overline{N} = 500$ peers.

We consider two models of peer churn. In the *Markovian churn* model the peer arrival process is a Poisson process with arrival intensity $\lambda(t)$, and the peer holding time distribution is exponential with mean $1/\mu$. In the *Flash-crowd churn* model N_f peers arrive according to a homogeneous Poisson process with intensity λ_f over a time interval $[t_f, t_f + N_f/\lambda_f]$, and the peers remain in the overlay until the end of the simulation.

B. Playout Adaptation in Steady State

The first scenario we consider is a system in steady state, generated using the Markovian churn model. The peer arrival rate is $\lambda(t) = 1.66/s$ and the mean holding time is $1/\mu = 300$ s. The average number of peers is thus $\overline{N} = 500$.

Fig. 1 shows the normalized histogram of the chunk missing ratio of the peers after a warm-up period of 2000 seconds in a simulation of 6000 seconds, for the *CA*, *DA* algorithms and without playout adaptation (*NA*). For *NA*, we used a fixed playback delay of $B = 12.5$ s, which is the average playback delay obtained using the *CA* algorithm and $\tau = 0.01$, $\eta = 0.5$. Fig. 2 shows the cumulative distribution function of the playback delay for the same time interval.

The results show that even in the case of a system in steady state, adaptation decreases the occurence of high chunk missing ratios (the tail of the histogram compared to the system without adaptation). Comparing the results for the *CA* and for the *DA* algorithm with different policies we observe that the *Fixed* policy not only leads to high chunk missing

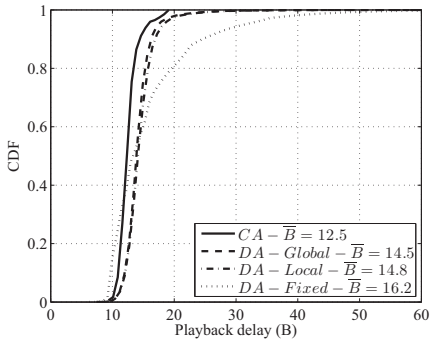

Fig. 2. CDF of the playback delays for the *CA* and the *DA* algorithms with different startup policies. The average playback delays are shown in the legend. The target miss ratio is $\tau = 0.01$ and $\eta = 0.5$. Overlay of $\overline{N} = 500$ peers.

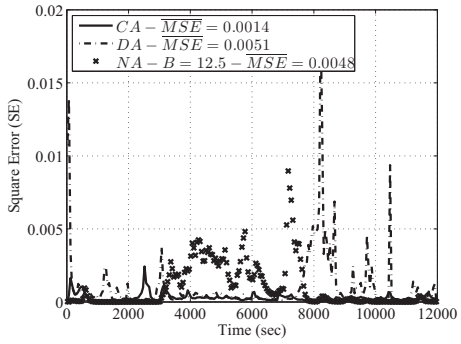

Fig. 3. SE of the chunk missing ratio. The target chunk miss ratio is $\tau = 0.01$ and $\eta = 0.5$ which means that the SE is 0 when $\pi(t) \in [0.005, 0.01]$. The MSE is also shown in the legend.

ratios (Fig. 1) but it also leads to high playback delays (Fig. 2). The *Global* and the *Local* policies, however, lead to lower chunk missing ratios than the *CA* algorithm at the price of higher playback delays. The reason is that using *DA* it is not the *average* chunk missing ratio that is maintained in the target band, but it is each *invididual* peer's chunk missing ratio.

C. Adaptation in a Non-stationary System

The second scenario we consider is a non-stationary system, generated using the Markovian churn model in the following way. The total simulation time is 12000 s, and the peer arrival intensity is $\lambda(t) = 1.66/s$ for the time intervals $[0, 3000)$ and $[7000, 12000]$, and it is $\lambda(t) = 16.66/s$ for the time interval $[3000, 7000)$. The average peer holding time is $1/\mu = 300$ s. Thus, there is a transition from an overlay of $\overline{N} = 500$ peers to an overlay of $\overline{N} = 5000$ peers, and then back to an overlay of $\overline{N} = 500$ peers.

Fig. 3 shows the square error (SE) calculated over consecutive intervals of 50 seconds for the *CA* and *DA* algorithms and without adaptation (*NA*). The figure also shows the mean square error ($\overline{MSE}$) for the entire simulation. The SE for the *CA* algorithm is very small, which means that the algorithm manages to keep the chunk missing ratio within the interval

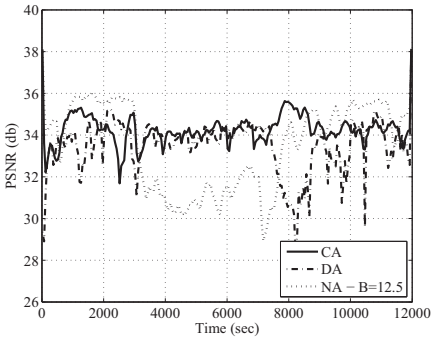

Fig. 4. PSNR of the received video over time for the *CA*, *DA* algorithms and for the an overlay without playout adaptation.

$[\eta\tau, \tau]$ with small deviations, despite the increase in the overlay size by a factor of ten. Without adaptation the chunk missing ratio increases significantly during the time period with high arrival intensity. We also notice an increase in the SE for the *DA* and *NA* systems right after the the switch from the high arrival intensity to the low arrival intensity period ($t = 7000$ s). For the system without adaptation the degradation is expected, since it takes some time for the peers to acquire new neighbors and maintain an adequate download rate.

For the system employing the *DA* algorithm, the explanation for the performance degradation lies in the algorithm itself. If peers do not change their playback delay in a coordinated manner, but based on local information, clusters of peers with similar playback delays emerge. When the departure rate in the overlay is higher than the arrival rate, peers that are losing their neighbors try to find new ones but the new neighbors may have different playback delays, i.e., only partially overlapping buffers, decreasing thus the efficiency of the data exchange. This is the reason why the same degradation is not observed when using the *CA* algorithm that keeps peers synchronized. It is therefore necessary to use some form of coordination among peers when performing playout adaptation in a distributed manner in order to maintain similar playback delays for all the peers in the overlay.

Next, we show the gain of playout adaptation in terms of the video distortion. We quantify the video distortion through the Peak Signal-to-Noise Ratio (PSNR) of the received video. The PSNR is defined as $PSNR = 10 \cdot log_{10}(255^2/D)$, where D is the total (source and channel) distortion. In order to calculate the distortion of the video as a function of the chunk missing ratio, we use the loss-distortion model presented in [18].

Fig. 4 shows the average PSNR over time, calculated for the Foreman sequence for the non-stationary system. During the initial low arrival intensity period the performance with and without adaptation is similar. When the overlay expands, the PSNR degrades significantly if no adaptation is used, the difference varying between 2 and 4 dB, compared to the overlays using adaptation. The *CA* algorithm manages to maintain the PSNR almost constant for the whole simulation

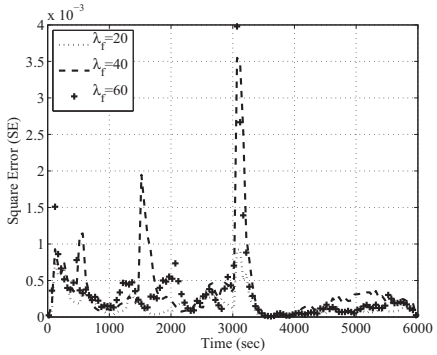

Fig. 5. SE of the chunk missing ratio for different arrival intensities during the flash crowd. The target chunk miss ratio is $\tau = 0.01$ and $\eta = 0.5$.

time and the transitions from the small to large overlay and back are indistinguishable. The *DA* algorithm performs well too but, as also shown earlier, its PSNR degrades sharply (up to 6 dB) after the transition from the high arrival intensity to the low arrival intensity period starts. Furthermore, the PSNR of the system employing the *DA* algorithm exhibits large variations after $t = 7000$ s, which can have a significant impact on the perceived video quality.

D. Adaptation under Flash Crowd Scenarios

Finally, we evaluate the adaptation algorithms under a flash crowd scenario. We study the transition from an overlay of $N_1 = 500$ peers to an overlay of $N_2 = 5000$ peers at different peer arrival intensities. We simulate the flash-crowd scenario by superposing two processes. The first process is generated using the *Markovian churn* model, with arrival rate $\lambda(t) = 1.66/s$ for the whole duration of the simulation. The peer holding times are exponentially distributed with mean $1/\mu = 300$ seconds. The second process is generated using the *Flash-crowd churn* model with $t_f = 3000$ s and arrival rates $\lambda_f = 20/s$, $\lambda_f = 40/s$ and $\lambda_f = 60/s$ to generate a small, moderate and severe flash crowd, respectively. The total simulation time is 6000 seconds.

Fig. 5 shows the square error (SE) of the chunk missing ratio calculated over consecutive intervals of 50 seconds. For the period before the flash crowd the SE lies in the same region as in Fig. 3. During the flash crowd though, there is a sharp increase in the SE, whose peak is higher with higher λ_f. The important thing to observe here is that the *CA* algorithm manages to converge to the desired band within a short period after the flash crowd. The time it takes to return the chunk missing ratio to the desired band does not seem to depend on λ_f, which indicates that the *CA* algorithm adapts well to different levels of impairment by appropriately increasing the adaptation step size.

VI. Conclusion

In this work we addressed the problem of playout adaptation in P2P streaming systems. We defined two algorithms that perform playout adaptation. Using coordinated adaptation peers adapt their playback delay synchronously, maintaining fully overlapping playout buffers. Using distributed adaptation each peer adapts its playback delay on its own, without any coordination with neighbors. We used extensive simulations to validate the algorithms and to evaluate their performance under various churn models. We showed that playout adaptation does not decrease the system's performance in steady state and that it is essential when the overlay size is changing. Our results indicate that peers can perform playout adaptation based solely on information about the playout position of their neighbors, which can be easily obtained locally. Nevertheless, our results show that distributed playout adaptation works well only if peers maintain similar playback delays across the whole overlay, thus some form of coordination is necessary.

References

[1] M.C. Yuang, Po L. Tien, and Shih T. Liang. Intelligent video smoother for multimedia communications. *IEEE Journal on Selected Areas in Communications*, 15(2):136 –146, February 1997.

[2] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *IEEE INFOCOM*, pages 680 –688, June 1994.

[3] I. Chatzidrossos, Gy. Dán, and V. Fodor. Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates. *P2P Networking and Applications*, 3:208–221, March 2010.

[4] Gy. Dán and V. Fodor. Delay asymptotics and scalability for peer-to-peer live streaming. *IEEE Transactions on Parallel and Distributed Systems*, 20(10):1499–1511, October 2009.

[5] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: a survey of playout schedulers. *IEEE Network*, 16(3):30 –40, May/June 2002.

[6] Werner Geyer, Christoph Bernhardt, and Ernst Biersack. A synchronization scheme for stored multimedia streams. In *Interactive Distributed Multimedia Systems and Services (IDMS)*, pages 277–295. Springer Verlag, 1996.

[7] M. Kalman, E. Steinbach, and B. Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):841–851, June 2004.

[8] E. Biersack, W. Geyer, and C. Bernhardt. Intra- and inter-stream synchronisation for stored multimedia streams. In *Proc. of the 3rd IEEE International Conference on Multimedia Computing and Systems*, pages 372 –381, June 1996.

[9] C. Vassilakis, N. Laoutaris, and I. Stavrakakis. On the impact of playout scheduling on the performance of peer-to-peer live streaming. *Computer Networks*, 53:456–469, March 2009.

[10] H. Jiang and S. Jin. NSYNC: Network synchronization for peer-to-peer streaming overlay construction. In *Proc. of NOSSDAV*, pages 1–6, 2006.

[11] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic live streaming: Optimal performance trade-offs. In *Proc. of ACM SIGMETRICS*, June 2008.

[12] A. Carta, M. Mellia, M. Meo, and S. Traverso. Efficient uplink bandwidth utilization in P2P-TV streaming systems. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, December 2010.

[13] Y.F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Modeling the impact of frame rate on perceptual quality of video. In *Proc. of the 15th IEEE International Conference on Image Processing (ICIP)*, pages 689–692, 2008.

[14] Computing TCP's retransmission timer - IETF RFC 2988, November 2000.

[15] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, pages 43–56, August 2000.

[16] P2PTV-SIM, http://napa-wine.eu/cgi-bin/twiki/view/public/p2ptvsim.

[17] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *Proc. of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1 –10, August 2010.

[18] V. Vukadinovic and Gy. Dán. Multicast scheduling for scalable video streaming in wireless networks. In *ACM Multimedia Systems (MMSys)*, February 2010.