Playback Policies for Live and On-Demand P2P Video Streaming

Fabio V. Hecht[1], Thomas Bocek[1],
Flávio Roberto Santos[1,2⋆], and Burkhard Stiller[1]

[1] University of Zurich, Communication Systems Group CSG, Zurich, Switzerland
[2] Federal University of Rio Grande do Sul, INF, Porto Alegre, Brazil
{hecht,bocek,santos,stiller}@ifi.uzh.ch

Abstract. Peer-to-peer (P2P) has become a popular mechanism for video distribution over the Internet, by allowing users to collaborate on locating and exchanging video blocks. The approach LiveShift supports further collaboration by enabling storage and a later redistribution of received blocks, thus, enabling time shifting and video-on-demand in an integrated manner. Video blocks, however, are not always downloaded quickly enough to be played back without interruptions. In such situations, the playback policy defines whether peers (a) stall the playback, waiting for blocks to be found and downloaded, or (b) skip them, losing information. Thus, for the fist time this paper investigates in a reproducible manner playback policies for P2P video streaming systems. A survey on currently-used playback policies shows that existing playback policies, required by any streaming system, have been defined almost arbitrarily, with a minimal scientific methodology applied. Based on this survey and on major characteristics of video streaming, a set of five distinct playback policies is formalized and implemented in LiveShift. Comparative evaluations outline the behavior of those policies under both under- and over-provisioned networks with respect to the playback lag experienced by users, the share of skipped blocks, and the share of sessions that fail. Finally, playback policies with most suitable characteristics for either live or on-demand scenarios are derived.

Keywords: P2P, live streaming, video on demand, playback policies

1 Introduction

The peer-to-peer (P2P) paradigm has been successfully applied to increase scalability and decrease cost for the publisher of video streams on the Internet [7]. Since users of a P2P system already collaborate on distributing video streams, LiveShift [6] makes further collaboration possible by allowing peers to store received video streams and distribute them in the future, thus enabling time shifting or – if the combined storage is large – even video-on-demand (VoD). This

⋆ Work developed while being a guest Ph.D. Student at University of Zurich, CSG@IFI.

gives users the freedom to, without having previously prepared any local recording, watch any program from an arbitrary position in the time scale, skipping uninteresting parts until seamlessly catching up with the live stream as desired.

Content availability in any video streaming system is affected by network and server conditions, possibly causing content not to be downloaded on time to be played. The challenge increases in P2P systems due to, *e.g.*, poorly managed networks, asymmetric bandwidth of peers, traffic-shaping at Internet service providers, free-riding, limited view of peers, and the fact that users change their interest frequently – switching channels and, in case of LiveShift, also time shifting. Content may even be available in some peers, but not downloaded before playback, because these peers have allocated all their upload capacity to serve other peers. In this paper, the term *content availability* is thus defined as content that is downloaded before its playback deadline.

The *playback policy* is the decision on, when content is unavailable, whether to stall playback, or skip to a block that has already been downloaded. Though any P2P video streaming system needs to implement a playback policy, current systems either omit this information, or adopt an arbitrarily-defined policy. Up to the authors' knowledge, this paper's work is the first aiming specifically at investigating and comparing the effect of different playback policies.

The two main research questions this paper addresses are (1) *do different playback policies affect user experience in a P2P video streaming system*, and (2) *which playback policies are most suitable for live and on-demand scenarios?* In order to answer these questions, this paper briefly overviews P2P video streaming and introduces key terminology in Sect. 2. Section 3 displays the related work survey on playback policies used by different live and on-demand P2P video streaming systems. Based on this, a classification and generalization of different playback policies are presented in Sect. 4, enabling a meaningful comparison among them. These policies have been implemented in LiveShift; Sect. 5 presents their evaluation and comparison under a variety of carefully-selected scenarios and parameters. Finally, Sect. 6 concludes this paper.

2 Background and Terminology

LiveShift, like most widely-deployed P2P video streaming systems, employs the mesh-pull approach [7, 10], which consists on dividing the stream into *blocks* that are announced and exchanged directly between peers with no fixed structure.

Figure 1 illustrates various terms used in this paper, and Table 1 defines nomenclature. In LiveShift, blocks have a fixed length L in the time scale. A viewing *session* starts when a user chooses a channel and starting time t_0 (the current time, if live streaming). While the user holds on to (*i.e.* watches) a channel, the system attempts to locate and download the corresponding blocks. Ideally, the user would experience *perfect playback*, that is, without interruptions, the block to be played $b_p(t)$ would be obtained based on the *playback position* t:

$$b_p(t) = t/L \tag{1}$$

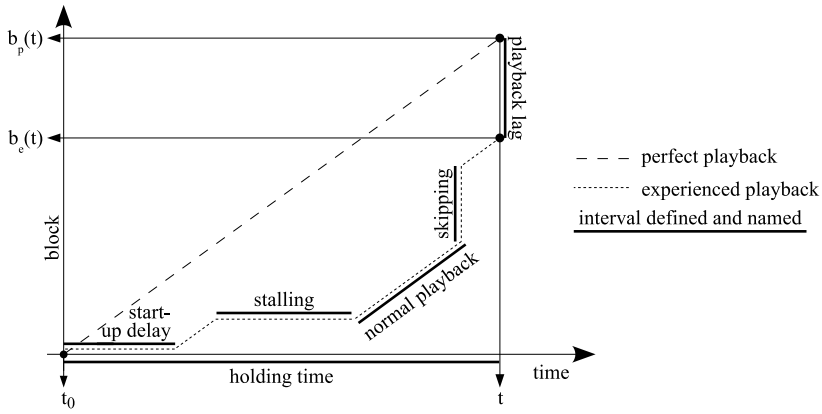

Fig. 1. Terminology

However, due to lack of content availability, the playback experienced by the user at t is the block $b_e(t)$, as given by (2), where $n_{sk}(t)$ is the number of skipped blocks and $t_{st}(t)$ is the time stalled from t_0 until t.

$$b_e(t) = n_{sk}(t) + (t - t_{st}(t))/L \tag{2}$$

Performing *initial buffering* corresponds to stalling playback until the playback buffer accumulates a number of blocks (typically a fixed configuration parameter), as an attempt to reduce the chance of skipping or stalling during playback. The *start-up delay* is the experienced stalling time caused by initial buffering. The *playback deadline* determines the time that a particular block is due to be played, according to the playback policy.

Table 1. Nomenclature

L	Block length (time unit)	ℓ	Buffer size (in blocks)
t_0	Session start time	α	Initial buffering coefficient
t	Current playback position	β	Stalling coefficient
$b_p(t)$	Block played at t if perfect playback	t_d	Remaining download time
$b_e(t)$	Block played at t	t_p	Remaining movie length
$t_{lag}(t)$	Playback lag at t	r	Relative incoming block rate
$n_{sk}(t)$	Skipped blocks from t_0 to t	T	Maximum retries
$n_{pl}(t)$	Played blocks from t_0 to t	n	Minimum block ratio

The term playback lag is commonly defined for live streaming as the elapsed time from the moment a block is generated at the source until it is played at the peer side [9]. In LiveShift, the concept of *playback lag* is extended also for viewing time-shifted streams; playback lag is thus defined by (3) as the time

difference between the block that is playing, according to the playback policy used, and the block that would be playing, if there were no interruptions from the moment the user pressed the play button. This extension in the definition of playback lag preserves its original concept of measuring the overall ability of the system of locating and downloading content, while being applicable to non-live viewing as well. *Liveness* is a general term that stands for low playback lag.

$$t_{lag}(t) = (b_p(t) - b_e(t)) * L \tag{3}$$

While both stalling and skipping negatively impact user-perceived video quality, their impact is different. On one hand, when stalling occurs, the video stream is interrupted and playback lag is increased. Besides, peers with higher playback lag lose their ability of providing streams with more liveness to other peers, negatively impacting the entire system. On the other hand, when skipping occurs, image quality might be impaired. Besides, since skipped blocks are not downloaded, they cannot be uploaded to any other peer, creating buffer "holes" that may harm the distribution overlay. In both cases, peers need a larger number of providers to compensate for those missing blocks, which may be challenging, since upload capacity is typically a rare resource in such systems [7].

3 Related Work

Though the implementation of a playback policy is required in any video streaming system, it is often omitted from their specification. Works that do describe the adopted playback policy are introduced in this section.

The most popular P2P live video streaming applications, such as SopCast [2], are proprietary and do not disclose in detail their policies. Measurement works [13], though, suggest that these systems, after performing initial buffering, employ a window that moves at constant speed, skipping all blocks that are not downloaded on time. The assumption is that, since streaming is live, maintaining liveness is more important than attempting to play every single block. It also helps keep peers mutually interested in a narrow content window. Such policy is also used on works that model live P2P systems [8].

For VoD, the assumption is frequently the opposite. Since liveness is not important, an intuitive policy would be, after performing initial buffering, stall every time a block to be played is missing. In a P2P system, though, such policy could cause playback to stall for a very long time in case there are a few very rare blocks. The VoD P2P client Tribler [11] addresses this issue by stalling playback only if less than 50% of a 10-second playback buffer is filled; otherwise, it skips to the next available piece.

The work presented in [12] also uses Tribler for VoD, but adopts a different policy. Playback is stalled until a 10-second-long buffer is filled and the remaining download time is less than the duration of the video plus 20%. The policy does not allow skipping.

Gridcast [4] is a P2P VoD system which playback policy consists on stalling if a block is not available at its playback deadline, while attempting to play it

up to 10 times. If the block still has not been downloaded, playback skips to the next block. Initial buffering is 10 seconds, and each block is 1 second long.

LiveShift [6] adopts a unified policy for live and on-demand streaming which consists on skipping n contiguous missing blocks if and only if the peer holds at least $2n$ contiguous blocks immediately after those, otherwise stalling. It aims not to let peers stall for long in case only a few blocks are not available from current neighbors, while skipping if there is a good chance of continuing playback without interruptions.

While these works discuss briefly the playback policy adopted, they do not offer a plausible proof or justification to why such algorithms and parameters were chosen in the foreseen scenarios.

4 Playback Policies

This section describes and generalizes a set of four playback policies based on the related work survey presented in Sect. 3, plus a new Catchup policy, enabling a fair and meaningful comparison among them. Analysis and discussion on respective trade-offs in both live and on-demand scenarios are as well performed.

4.1 Always Skip and Skip/Stall Policies

The Always Skip policy, commonly used for live streaming, consists on always skipping missing blocks to maintain liveness. It is defined by $P_{as} = (\ell, \alpha)$, where ℓ represents the size of the playback buffer (in blocks), and $\alpha \in (0, 1]$ corresponds to the share of blocks that the buffer must hold before starting playback. After the first block has been played, buffered blocks are attempted to be played sequentially and at constant speed. Missing blocks are immediately skipped; however, if the buffer is empty, playback stalls to perform again initial buffering. This is done so peers adapt their playback position $b_e(t)$ according to the blocks that can be found at – and downloaded from – currently known peers.

The Skip/Stall (`sk`) policy is an extension to the Skip policy to allow stalling as in Tribler [11]. It is defined as $P_{sk} = (\ell, \alpha, \beta)$, which introduces the $\beta \in [0, 1]$ coefficient, such that, when a block at $b_e(t)$ is missing and the buffer is not empty, the system stalls playback until a share β of the playback buffer is filled; then, it skips to the next available block. The Always Skip policy is, thus, an instance of the `sk` policy when $\beta = 0$.

4.2 Remaining Download Time Policy

Especially for VoD, it is reasonable to define a playback policy that depends on the remaining download time, so stalling is reduced for users with a fast network, while buffering increases with slower networks. The Remaining Download Time (`rd`) policy stalls playback until the remaining download time $t_d \leq t_p$, the remaining movie length. In order to apply this policy to LiveShift, the concept of remaining playback time must be defined, since the stream ahead of playback

position may be very large – it extends until the current (live) time. Hence, t_p is a parameter that may be set to, *e.g.*, 30 minutes of video that buffering will attempt to guarantee playback for.

The `rd` policy can be modeled as $P_{rd} = (\ell, \alpha, \beta, t_p)$ by using the same algorithm as defined for the `sk` policy, but using a variable buffer size ℓ', calculated based on the parameters t_p and ℓ, instead of ℓ directly. Infinite geometric series are used to calculate how long the playback buffer would last, since the application continues to download blocks while the buffer is being used. If i represents the incoming block (*i.e.* download) rate, and L being block length, let r represent the relative incoming block rate, such that $r = i * L$; thus, if $r = 1$, the peer is downloading blocks at a rate exactly enough to keep normal playback. The variable buffer size ℓ' can therefore be calculated as shown in (4), where ℓ is used both for initial buffering and as a general lower limit of the buffer size (when, for example, $r \geq 1$). The coefficient α is present in the equation to preserve the semantic of remaining download time, since only a share α of the buffer is required by the playback policy to be held in the buffer.

$$\ell' = \max\left(\frac{t_p}{L} \times \frac{(1-r)}{\alpha}, \ell\right) \tag{4}$$

4.3 Retry Policy

The Retry (`re`) playback policy is similar to the policy implemented in Gridcast [4], and is defined as $P_{re} = (\ell, \alpha, T)$. It consists on performing initial buffering, then stalling if the block at playback position t is not available. The system retries playing the missing block up to T times, which brings playback to stall for a maximum of $T * L$ seconds. As soon as the missing block is downloaded, it will be played back; if the stalling threshold is hit, though, playback skips to the next available block.

4.4 Ratio Policy

The Ratio (`ra`) policy aims at skipping blocks only if there is a high chance of then continuing playback without interruptions. It is described as $P_{ra} = (\ell, \alpha, n)$, where ℓ and α retain their previous meaning. After initial buffering, if the block at playback position is locally held, it is always played. If, however, the block is missing, a ratio $1 : n$ is given, such that x contiguous missing blocks are skipped, if and only if, at least xn contiguous blocks are held directly after those.

4.5 Catchup Policy

The Catchup (`ca`) playback policy is introduced to keep playback lag very low at the cost of skipping more blocks than other policies. It is defined by $P_{ca} = (\ell, \alpha)$, where ℓ and α are used to perform initial buffering as in the `sk` policy. After playback has started, all missing blocks are skipped, as long as the buffer is not empty. When it is indeed empty, playback position is restored to the original one

by skipping all blocks responsible for playback lag until $b_p(t) = b_e(t)$. It is meant to provide a practical limit on the lowest possible playback lag achievable.

5 Evaluation

All playback policies defined in Sect. 4 have been implemented into LiveShift. Experiments were conducted using the entire LiveShift code in a deployed test-bed of 20 machines [3]. The main objective was to compare how different playback policies affect user experience of a P2P video streaming system under scenarios with different levels of content availability.

Table 2 displays those different scenarios used. Peers are divided in classes according to their maximum upload capacities – while high upload capacity (HU) peers and peercasters (PC) are able to upload at a rate equivalent to 5 times the bit rate of the video stream being transmitted, low upload capacity (LU) peers are able to upload at only 0.5 times the original stream rate. The increasing number of LU peers causes available upload bandwidth to decrease; while Scenario s1 has abundance of total upload capacity compared to the number of peers to be served, in Scenario s4, the chance that peers experience content unavailability is much higher. It is important to note that peers that have unused upload capacity might only hold unpopular content, leading to suboptimal overlay resource usage. Peers are not artificially limited in download bandwidth, and latency between peers was introduced by using a random sample from the King dataset [5], and enforced using DummyNet [1]; the sample contains an average latency of 114.2 ms. This paper only displays results for scenarios s1 and s4, for brevity and because scenarios s2 and s3 produced expected results between s1 and s4.

Table 2. Evaluation scenarios

Scenario	*# PC*	*# HU*	*# LU*	*Total Peers*	*Total Upload Capacity*
s1	6	15	60	81	135
s2	6	15	90	111	150
s3	6	15	120	141	165
s4	6	15	150	171	180

Multiple instances of LiveShift were executed using the same settings as in [6]. In short, peers were created with an inter-arrival time of 1 s. Every peer was programmed to repeatedly switch to a channel and starting time t_0, then hold to the channel, attempting to locate and download blocks. While holding to the channel, every peer reported, once per second, its experienced playback lag $t_{lag}(t)$, as defined in Sect. 2, and share of skipped blocks $n_{sk}(t)/(n_{pl}(t) + n_{sk}(t))$. Channel popularity and holding time were both characterized by traces, as described in [6]. Results were obtained through 10 runs of 20 minutes each.

Due to content unavailability, a peer may sometimes experience very long stalling times. In such cases, it is not realistic to assume that users would wait indefinitely for the content. Thus, when a peer is able to play less than 50% in a moving window of 30 seconds, playback is considered *failed*, that is, the user is considered to have given up and switched to another $(channel, t_0)$. Buffering is not taken into account, since it is part of the playback policy being investigated.

Since the goal of the evaluation is to measure the impact of playback policies in the entire overlay, on each run, all peers were configured to adopt one of the playback policies defined in Sect. 4. Each playback policy, as specified in Table 3, has been investigated using different values for their main parameters, based on values seen in the literature, complemented by additional values that allow a deeper understanding of their effect. To make results better comparable, parameters that apply to all playback policies were kept constant; all experiments were obtained using $\ell = 6\,\text{s}$, $\alpha = 0.8$, and $L = 1\,\text{s}$. The evaluation metrics used are playback lag, share of skipped blocks, and share of failed playback sessions.

Table 3. Playback policies and parameters

Policy	*Parameter*	*Identifier*
Skip/Stall	$\beta = 0$	`sk-0`
	$\beta = .5$	`sk-.5`
	$\beta = .75$	`sk-.75`
Remaining Download Time	$t_p = 5\,\text{s}$	`rd-5`
	$t_p = 30\,\text{s}$	`rd-30`
	$t_p = 60\,\text{s}$	`rd-60`
Retry	$T = 1$	`re-1`
	$T = 5$	`re-5`
	$T = 10$	`re-10`
Ratio	$s = 2$	`ra-2`
	$s = 3$	`ra-3`
	$s = 5$	`ra-5`
Catchup	(none)	`ca`

5.1 Playback Lag

Playback lag is an important metric to evaluate user experience, since a lower value denotes a lower start-up delay, less interruptions, and more closeness to what the user initially intended to watch. It is expected to increase with larger sessions, as well as with lower content availability, due to stalling. Reports from each peer and run were collected and an average was calculated for every 1-minute interval. The same proceeding was performed on all runs.

The distribution of playback lag among different peers and at different t values was analyzed for each policy. For most peers, playback lag differences for

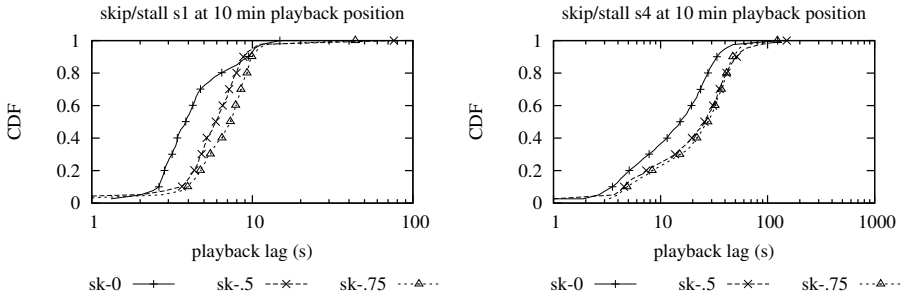

Fig. 2. CDF of playback lag under Skip/Stall playback policy in scenarios s1 and s4

the investigated parameters are consistent, as exemplified in Fig. 2, which shows the CDF of playback lag at 10 min of holding time under the `sk` policy. Peers, however, with highest playback lag (*i.e.* above 90$^{\text{th}}$ percentile) suffer from severe content unavailability, as a result of the high channel switching frequency in the defined peer behavior; these peers are not able to download any blocks, so the playback policy cannot play a significant role. Since this occurs as well under all other investigated policies, all playback lag plots on the rest of this section focus on the 80$^{\text{th}}$ percentile, that is, the maximum playback lag experienced by 80% of the peers.

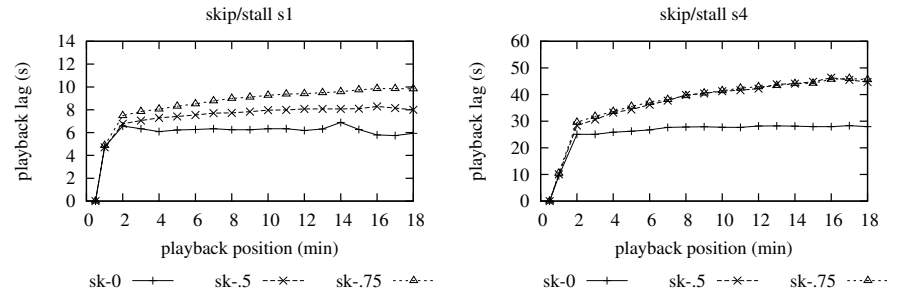

Fig. 3. Playback lag under Skip/Stall playback policy in scenarios s1 and s4

Always Skip and Skip/Stall Playback Policies. Experiments with the `sk` playback policy were made using three different values for the parameter β, as shown in Fig. 3. While the x-axis represents the time t (in minutes) for which a user holds on to a channel, the y-axis represents the playback lag $t_{lag}(t)$ (in seconds). The experiments reveal that the `sk` playback policy is extremely flexible. It is able to maintain a relatively low playback lag even for longer

sessions (higher holding time values) when $\beta = 0$ (`sk-0`, the Always Skip policy). In Scenario s1, `sk-.5` and `sk-.75` display very distinct results, yet on Scenario s4 they yield very similar results in terms of playback lag. This is due to the fact that, in a scenario with more available upload bandwidth, peers have more opportunity to perform several parallel downloads, hence the chance that a peer is able to download blocks out of order (thus being able to skip) is higher.

Remaining Download Time Playback Policy. The `rd` playback policy was instantiated with different values of t_p, which is the minimum playback time the policy attempts to guarantee playback for, considering the current download rate. Figure 4 shows that, on the over-provisioned Scenario s1, results differ little with the different parameters evaluated. This is due to the fact that peers can often download at a rate $r \geq 1$, therefore ℓ' frequently reaches its minimum value ℓ, as ℓ' decreases with a higher download rate. In the more bandwidth-restricted Scenario s4, larger values of t_p cause higher playback lag with higher holding times, as expected. In comparison with other playback policies, the Remaining Download Time playback policy shows the highest playback lag, which is due to a potentially larger buffer, especially with lower download rates.

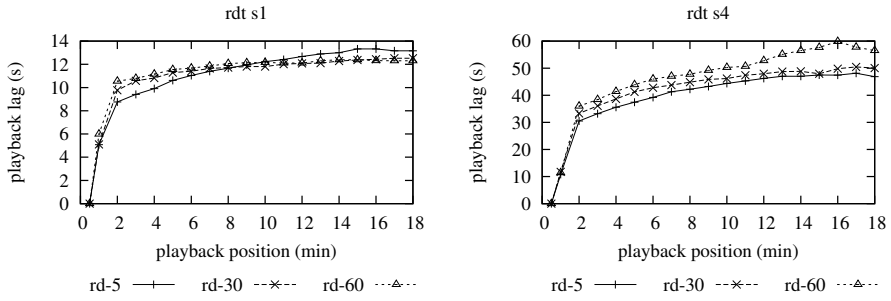

Fig. 4. Playback lag under Remaining Download Time playback policy in s1 and s4

Retry Playback Policy. The `re` playback policy was investigated with different values for the parameter T, which expresses the stalling limit per block. Figure 5 shows that, on both scenarios s1 and s4, while re-1 displays a lower playback lag than `re-5` and `re-10`, it is still higher than levels achieved under the `sk` policy. The fact that playback lag under `re-5` and `re-10` policies are very similar is due to the unlikelihood, on both s1 and s4, of situations in which playback needs to stall for longer than 5, but less than 10 seconds, until the block at playback position is downloaded.

Ratio Playback Policy. Results with the `ra` playback policy were obtained using different values for the parameter n. Figure 6 shows that it has a noticeable

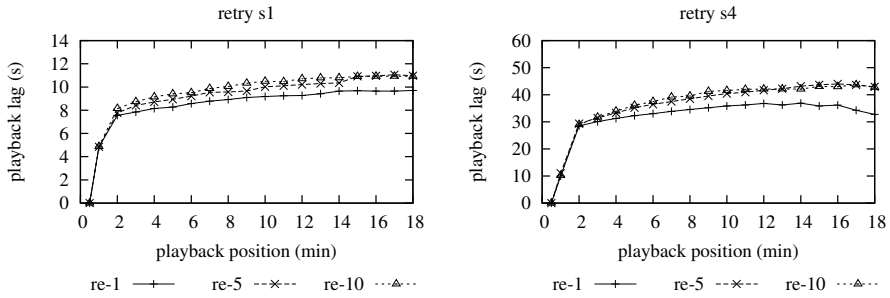

Fig. 5. Playback lag under Retry playback policy in scenarios s1 and s4

impact on the experienced playback lag in the over-provisioned Scenario s1, but not on s4. Like with the `sk` policy, peers have much fewer opportunity to skip in s4 due to the lower probability of performing parallel downloads.

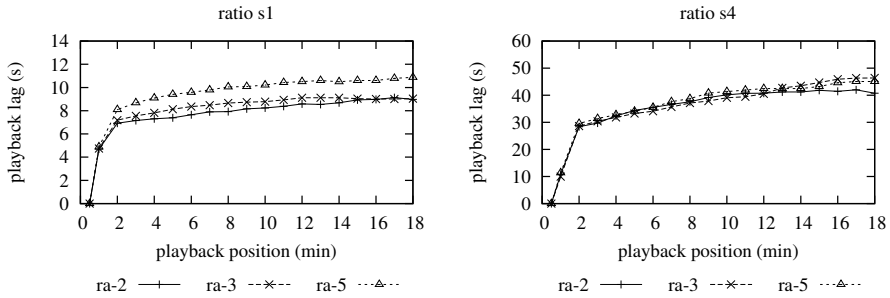

Fig. 6. Playback lag under Ratio playback policy in scenarios s1 and s4

Catchup Playback Policy. The Catchup (`ca`) playback policy is designed to keep a very low playback lag by resetting it to zero when the playback buffer is empty by skipping the necessary amount of blocks. Results show on Fig. 7 that, as designed, it displays a relatively lower playback lag in comparison to the other policies. Interestingly, while it displays in Scenario s1 a clearly higher playback lag than `sk-0`, the opposite is observed in s4. This happens due to the much higher probability in Scenario s4 that the buffer becomes empty and the catchup mechanism is therefore triggered.

5.2 Skipped Blocks

Figure 8 compares the mean share of skipped blocks under the playback policies and parameters investigated. Error bars indicate 95% confidence intervals of the

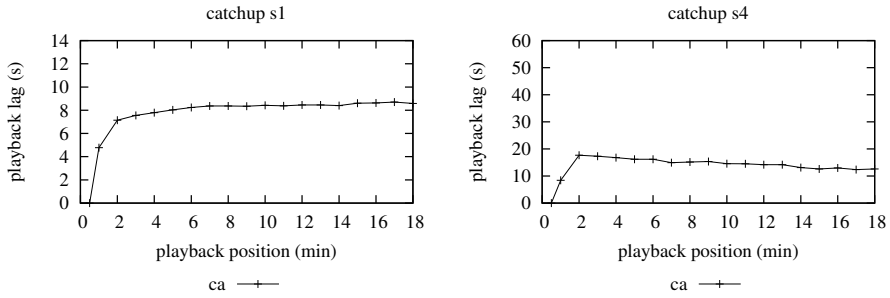

Fig. 7. Playback lag under Catchup playback policy in scenarios s1 and s4

means. The share of skipped blocks is, as expected, inversely proportional to the playback lag shown by each policy, hence `sk-0` and `ca` policies skip more blocks than other policies.

User-experienced image degradation levels vary according to specific video encoding and decoding algorithms (codecs) used – to which LiveShift is agnostic. Understanding both expected levels of skipped blocks and codec characteristics are thus crucial when choosing the appropriate playback policy for a specific situation.

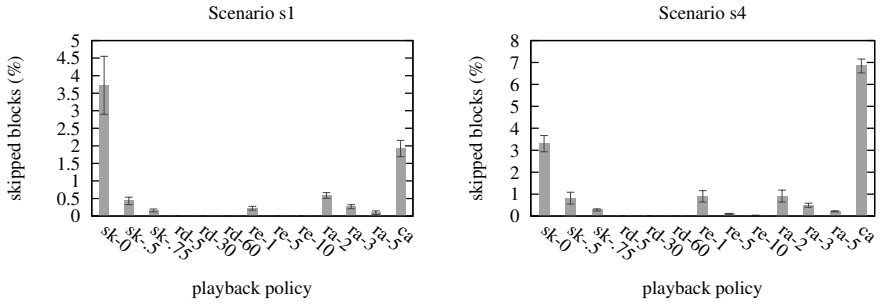

Fig. 8. Skipped blocks in scenarios s1 and s4

5.3 Failed Playback Sessions

The share of sessions in which the peer stalls for such a long time that playback is considered failed represent less than 0.5% of all sessions in Scenario s1, as shown in Fig. 9. In contrast, in Scenario s4, the mean oscillates between 9.5% and 13.5% in all scenarios. The overlapping 95% confidence interval error bars indicate that the share of failed playback depends rather on each scenario's available upload capacity than on the playback policy used.

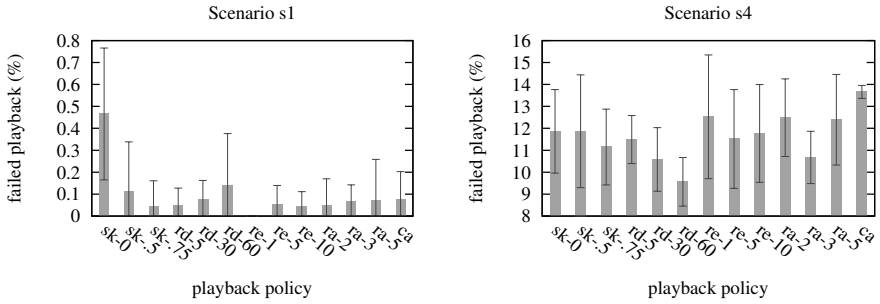

Fig. 9. Failed playback in scenarios s1 and s4

6 Discussion and Conclusions

Having observed the behavior of LiveShift under different playback policies, with different parameters, in scenarios ranging from under- to over-provisioned P2P networks, it is evident that *different playback policies* do *affect user experience in a P2P video streaming system*, in terms of playback lag and share of skipped blocks. Understanding their behavior is, hence, imperative to select the most appropriate policy for the desired result, whether it is keeping playback lag as low as possible, avoiding skipping many video blocks, or achieving a compromise. The ultimate decision may depend on the type of content being transmitted, or be left completely up to the user.

This raises the second research question, *which playback policies are more suitable for live and on-demand scenarios?* Under circumstances in which minimizing playback lag is the main goal, which might be desirable by viewers of live (*e.g.*, sports) events, the `ca` and `sk-0` are the most suitable policies studied, considering that they have consistently shown much lower playback lag for a majority of peers compared to all other approaches. This comes, however, at the cost of a higher number of skipped blocks. If lowest number of skipped blocks is the objective, the policies that have shown to skip less than 0.5% of the total blocks on both scenarios s1 and s4 are `sk-.75`, `re-5`, `re-10`, `ra-3`, `ra-5`, and all rd policies. These policies may be applied in cases in which occasional interruptions are of less importance than skipping content, for instance for VoD. Alternatively, compromising playback lag and skipped block rate may be the goal. Policies that show a skipped block rate inferior to 0.5% and playback lag inferior or equal to 45 seconds for 15-minute-long sessions (for 80% of peers) are, on the under-provisioned Scenario s4, the following: `ra-3`, `ra-5`, `re-5`, `re-10`, and `sk-.75`. In Scenario s1, the `ra-3`, `re-1`, `sk-.5`, and `sk-.75` are policies that yield a skipped block rate inferior to 0.5% and playback lag less than or equal to 9 seconds for 10-minute-long sessions, also for 80% of peers.

In all evaluated scenarios, peers adopt a uniform playback policy, which allows for an evaluation of their effect on the entire distribution overlay, if assumed that all users are interested in either live or on-demand characteristics. Future

work will investigate scenarios in which peers adopt mixed policies, which are likely in LiveShift. There is also the opportunity of combining characteristics of different policies. A further promising possibility is creating a predictive playback policy that considers past peer experiences to avoid stalling when the probability that a missing block is downloaded in a timely fashion is low.

Acknowledgments

This work has been performed partially in the framework of the of the European FP7 STREP SmoothIT (FP7-2008-ICT-216259) and was backed partially by the FP7 CSA SESERV (FP7-2009-ICT-258138) coordination inputs on incentives.

References

1. Dummynet - Network Emulation Tool for Testing Networking Protocols. http://info.iet.unipi.it/ luigi/dummynet/, last visited: Februray 2012.
2. SopCast - Free P2P Internet TV. http://www.sopcast.org, last visited: 07.12.2011.
3. Test-bed Infrastructure for Research Activities – Communication Systems Group (CSG) at the Department of Informatics (IFI), University of Zurich (UZH). http://www.csg.uzh.ch/services/testbed/, last visited: 07.12.2011.
4. B. Cheng, L. Stein, H. Jin, X. Liao, and Z. Zhang. GridCast: Improving Peer Sharing for P2P VoD. *ACM Transactions on Multimedia Computing, Communications and Applications*, 4:26:1–26:31, Nov. 2008.
5. K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating Latency Between Arbitrary Internet End Hosts. In *2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 5–18, New York, NY, USA, Nov. 2002.
6. F. Hecht, T. Bocek, R. G. Clegg, R. Landa, D. Hausheer, and B. Stiller. LiveShift: Mesh-Pull P2P Live and Time-Shifted Video Streaming. In *36th IEEE Conf. on Local Computer Networks*, LCN 2011, pages 319–327, Bonn, Germany, Oct. 2011.
7. X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8):1672–1687, Dec. 2007.
8. R. Kumar, Y. Liu, and K. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *26th IEEE International Conference on Computer Communications*, INFOCOM 2007, pages 919–927, May 2007.
9. C. Liang, Y. Guo, and Y. Liu. Is Random Scheduling Sufficient in P2P Video Streaming? In *28th International Conference on Distributed Computing Systems*, ICDCS '08, pages 53 –60, June 2008.
10. N. Magharei and R. Rejaie. Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In *Proceedings of IEEE INFOCOM 2007*, pages 1424–1432, 2007.
11. J. Mol. *Free-riding, Resilient Video Streaming in Peer-to-Peer Networks*. PhD thesis, Delft University of Technology, Jan. 2010.
12. J. J. D. Mol, A. Bakker, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips. The Design and Deployment of a BitTorrent Live Video Streaming Solution. *Intl. Symposium on Multimedia*, pages 342–349, Dec. 2009.
13. A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari. Will IPTV Ride the Peer-to-Peer Stream? *IEEE Comm. Magazine*, 45(6):86 –92, June 2007.