2008

Partial Forwarding Scheme for Dynamic Window Resizing in Live P2P Streaming Systems

Zhipeng Ouyang
University of Nebraska - Lincoln

Lisong Xu
University of Nebraska - Lincoln, xu@cse.unl.edu

Byrav Ramamurthy
University of Nebraska - Lincoln, bramamurthy2@unl.edu

IEEE Global Telecommunications Conference, 2008.
Digital Object Identifier: 10.1109/GLOCOM.2008.ECP.440
Publication Year: 2008 , Page(s): 1 - 6

A Partial Forwarding Scheme for Dynamic Window Resizing in Live P2P Streaming Systems

Zhipeng Ouyang, Lisong Xu and Byrav Ramamurthy
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0115, U.S.A.
Email: {zouyang, xu, byrav}@cse.unl.edu

***Abstract*—Peer-to-peer (P2P) streaming systems, in which individual nodes or peers operated by ordinary Internet users collaborate to serve video streams, have recently aroused considerable interest in both academia and industry. An important problem in P2P streaming systems is how to reduce their consumed bandwidth, which is a major concern of Internet service providers. Our work is motivated by the fact that a user may dynamically change the size of a window displaying a video stream according to his/her personal choice, a scenario we refer to as *dynamic window resizing*. In this paper, we propose a scheme called the Partial Forwarding Scheme (PFS) based on layered coding, in which users with small windows help in forwarding a part of the enhancement layer. PFS significantly reduces the total consumed bandwidth while still maintaining the desired streaming quality. Our extensive simulation results show that PFS can reduce the total consumed bandwidth by up to 40% while still maintaining satisfactory streaming quality.**

I. Introduction

Peer-to-peer (P2P) streaming systems, in which individual nodes or peers operated by ordinary Internet users collaborate to serve video streams, have recently aroused considerable interest in both academia and industry [11] [9]. It has been reported [15] [16] that current P2P streaming systems are able to support more than 500,000 concurrent users watching a live TV program with a relatively high streaming rate (above 300 Kbps). As P2P streaming systems have the potential to make any TV channel globally available, it is expected that the population of P2P streaming users will continue to increase.

For a user in a P2P streaming system, the size of a window displaying a video stream on his/her screen can be dynamically customized by the user, a scenario we refer to as dynamic window resizing. For example, a user can sometimes maximize the window size for a better view quality, and sometimes reduce the window size so that he/she can temporarily check other information such as real-time stock charts on the other area of the screen. Furthermore, a user may simultaneously watch multiple windows for multiple video streams, and different windows may have different sizes and different positions according to the user's preferences. For example, a user who has paid for the live broadcast of a football game is able to not only customize his/her screen with multiple windows displaying different views of the game, but also dynamically enlarge the window showing a specific view that attracts him/her the most at that moment.

The work reported in this paper is supported in part by UNL Layman Fund Award.

Motivated by issue of dynamic window resizing in a P2P streaming system, in this paper, we propose a scheme, called the Partial Forwarding Scheme (PFS), to provide a user with satisfactory streaming quality in case of dynamic window resizing, while at the same time significantly reducing the total consumed bandwidth, which is a major concern of Internet service providers. In most (if not all) of the current P2P streaming systems [15] [16], different users request the same video stream and display it in their windows, even though they may have different window sizes. In order to reduce the consumed bandwidth, our proposed PFS is based on layered coding [3] where a stream consists of a base layer and one or multiple enhancement layers. The idea of applying layered coding techniques in P2P streaming systems is not new [10] [2] [1]. The unique feature of PFS is that with PFS a user with a small window size participates in the packet forwarding of not only the base layer, but also part ($1/X$ portion) of the enhancement layer. To study the performance of PFS, we also consider two special cases of PFS as the reference protocols: the Bandwidth-First Scheme (BFS) where a user with a small window size does not forward any part of the enhancement layer (i.e. $1/X \approx 0$), and the Quality-First Scheme (QFS) where a small window forwards all of the enhancement layer (i.e. $1/X = 1$). BFS requires the minimum amount of bandwidth, whereas QFS can achieve the best streaming quality in case of dynamic window resizing. Our extensive simulation results show that PFS with a small value of $1/X$ can provide satisfactory streaming quality comparable to that provided by QFS, while at the same time requiring significantly reduced bandwidth comparable to that required by BFS.

The rest of this paper is organized as follows: Section 2 summarizes the related work, Section 3 describes P2P streaming systems and the dynamic window resizing problem. Section 4 describes our proposed PFS scheme to support dynamic window resizing. Section 5 presents our simulation results, and finally conclusions and future work are provided in Section 6.

II. Related Work

P2P streaming has been extensively studied recently, but to the best of our knowledge, the topic of dynamic window resizing has not been addressed. The most related work is P2P streaming with adaptive streaming, which is proposed to deal with heterogeneity and dynamics of P2P multimedia systems. Liu *et al.* [10] described a distributed incentive mechanism for mesh-pull P2P streaming networks with layered coding. Padmanabhan *et al.* [13] proposed an approach to distribute live streaming multimedia content to a potentially large and highly dynamic population of hosts. This is used to alleviate a flash crowd at a live streaming server. In this paper we study how to alleviate the impact of dynamic window resizing on the peers and their neighbors. Padmanabhan *et al.* [13] used a centralized tree management protocol and feedback mechanism; in contrast our study focuses on end users. Chakareski *et al.* [1] studied the performance of layered coding with and without path diversity, and concluded that layered coding provides good performance when transmission schedules could be optimized in a rate-distortion sense.

To improve the throughput of P2P streaming systems, several scheduling methods have been proposed, such as a stream-level method [2] and a block level method [17]. The former uses layered coding and determines routes for streams of different layers, while the latter is more fine-grained because it describes how a node fetches a particular block from a particular neighboring node.

III. Dynamic Window Resizing

Since the screen size at a user's computer monitor/display is limited, the window size used to show the streaming content and the layout of all windows must be selected to accommodate the user's preference and to maximize the screen utilization.

Figure 1 shows a simple example of a P2P streaming system, where a channel may be viewed using different window sizes. Each rectangle represents a peer, and the number on the top-left corner indicates the peer's ID, an uppercase letter in the center represents a channel being viewed in a large window, and a lowercase letter in the center represents a channel being viewed in a small window. For example, peer 1 is watching channel A in a large window (denoted by A in the figure) and at the same time peer 3 is watching channel A in a small window (denoted by a).

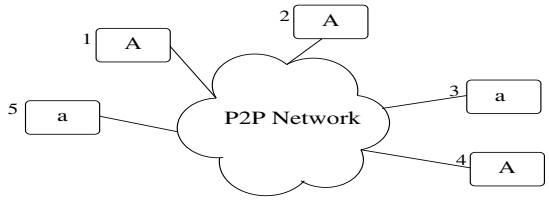

Fig. 1. Snapshot of the window sizes in a P2P streaming system, where a user can watch a channel with different window sizes.

As aforementioned, a user in a P2P streaming system can watch a channel in a large window, and then later change the window size from large to small, or vice versa. Please note that we use the term peers to refer to both nodes and users at these nodes throughout the paper.

We call these instances *Dynamic Window Resizing*, since the peer is still watching the same channel but with different window sizes, and it can dynamically change the window size of a channel at any time. Dynamic window resizing has more stringent requirements for the resizing delay, since the peer is already watching the channel, and thus expects very little or no delay when switching between different window sizes of the channel. In contrast, when a peer switches to a new channel, it is usually acceptable for him/her to wait for some reasonable amount of time before the channel starts playback on the screen.

IV. Schemes for Dynamic Window Resizing

In this section, we propose a scheme, called Partial Forwarding Scheme (PFS), to significantly reduce the total consumed bandwidth while at the same time maintaining the desired streaming quality. We compare the performance of PFS with two other straightforward schemes: Bandwidth-First Scheme (BFS) and Quality-First Scheme (QFS). All three schemes are studied and evaluated according to the following three design goals: 1) The resizing delay of a peer changing its window size from small to large should be very short. 2) A peer changing its window size from large to small should have little impact on its neighbor peers. 3) The total consumed bandwidth should not be very high.

A. Framework for Solving the Dynamic Resizing Problem

Our proposed PFS scheme is based on video coding techniques. There are two general coding techniques [7]: layered coding and multiple description coding (MDC). MDC sacrifices some compression efficiency to gain robustness to the loss of descriptions [4]. Since we want to design a scheme with less bandwidth requirement and MDC has too much redundancy, we choose to use layered coding. To simplify the explanation, we consider only a layered coding technique with a base layer and an enhancement layer. But our proposed PFS scheme can be easily extended to the cases with multiple enhancement layers. The base layer alone corresponds to a low resolution stream, and both the base layer and the enhancement layer together correspond to a high resolution stream. Different window sizes have different widths and/or lengths (in terms of pixels). When the window size of a user is small enough, the base layer alone can provide the user with satisfactory streaming quality; otherwise if the window size of a user is large, both the base layer and the enhancement layer are required to provide satisfactory streaming quality.

Our proposed PFS scheme does not require any specific P2P overlay architectures or packet scheduling algorithms. To simplify the description, below we consider a mesh-based P2P overlay topology [12] and the random packet scheduling algorithm [14] for live streaming, since both have been widely used in P2P live streaming systems.

978-1-4244-2324-8/08/$25.00 © 2008 IEEE.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

There are some solutions for shortening delays and/or saving bandwidth in P2P streaming systems, such as speeding up the content search in P2P systems [5] and quality adaptive method [6], but these solutions are not very effective here. First, every peer in this problem can arbitrarily choose its window size, so it is hard to maintain simple yet dynamic data structures for high speed searching. Second, quality adaptive method is proposed to deal with the dynamic behavior of the Internet's transmission resources, but what we consider here is about the end user's behavior.

B. Bandwidth-First Scheme (BFS)

In BFS, each peer only requests the necessary layers. A peer watching a channel receives either the base layer or both layers depending on its window size or resolution. For example, in Figure 1, peers 1, 2, and 4 all watch channel A at a high resolution in a large window (we refer to these peers as large-window peers), and thus they all receive both the base layer and the enhancement layer, while peers 3 and 5 watch channel A at a low resolution in a small window, and thus they receive only the base layer (we refer to these peers as small-window peers).

Now let us consider the three design goals in the following two possible window resizing cases.

First, peer 1 changes channel A from a large window to a small window. In this case, peer 1 just stops requesting the enhancement layer from its neighbors, but it still requests the base layer from them. Therefore, peer 1 can resize to a small window smoothly. However, since peer 1 does not have the enhancement layer any more, if any of its neighbors is watching channel A in a large window and is requesting the enhancement layer from peer 1, then a glitch may occur, as it takes some time for these neighbors to find a substitute for peer 1.

Second, peer 3 changes channel A from a small window to a large window. In this case, peer 3 must request the enhancement layer from its neighbors. If its neighbors do not have the enhancement layer or they do not have enough upload capacity for forwarding, peer 3 has to find new neighbors with the enhancement layer and that may take a long time. Therefore, peer 3 may experience a long delay. Since peer 3 still has the base layer, its resolution switching does not have any impact on its neighboring peers.

In both cases, since a peer only requests the necessary layers according to its window size, no bandwidth is wasted.

C. Quality-First Scheme (QFS)

In the QFS, each peer requests both the base layer and the enhancement layer independent of its window size or resolution. For example, in Figure 1, all peers receive both the base layer and the enhancement layer.

Now let us consider the three design goals in the two window resizing cases. In both cases, data forwarding is not affected by resizing, so no resizing delay and minimum impact on neighbors are caused. Since a peer requests both layers without considering the window size, the bandwidth of small-window peers used for the enhancement layer is wasted.

D. Partial Forwarding Scheme (PFS)

In PFS, we make small-window peers forward different enhancement layer portions as allowed by their upload bandwidths. The enhancement layer is divided into several portions, and each small-window peer forwards one of them. As shown in Figure 3, peers with large windows act the same as in BFS and in QFS. All small-window peers are divided into X groups according to the random numbers between 1 and X generated by the peers themselves when they join the system. Every peer in each group forwards $1/X$ portion of enhancement layer as shown in Figure 2, and X small-window peers from different groups together can provide the whole enhancement layer.

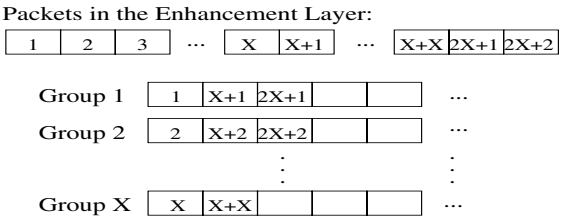

Fig. 2. An example of enhancement layer decomposition.

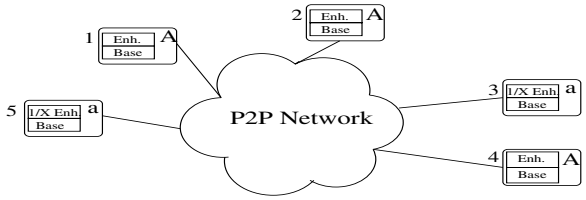

Fig. 3. PFS based on layered coding, where a small-window peer receives the base layer and some portion of enhancement layer, and a large-window peer receives both the base layer and enhancement layer.

Consider the design goals for the two cases that we discussed in Sections IV-B and IV-C. In case 1, peer 1 reduces requests for the enhancement layer from 100% to $1/X$, and maintains the base layer stream. Resizing for peer 1 is smooth. As peer 1 only has the base layer and $1/X$ portion of enhancement layer, its large-window neighbors which request the enhancement layer outside of this $1/X$ portion from peer 1 will experience an interruption. Compared with BFS, the impact of resizing can be alleviated here due to two reasons: first, the large-window neighbors requesting enhancement layer within this $1/X$ portion will not be affected. Second, in BFS, only large-window peers can help the affected peers while in PFS, if upload bandwidths are enough, small-window peers with different $1/X$ portions of the enhancement layer can support the affected peers as well.

In the second case, peer 3's resizing delay is shorter than the delay in BFS for the following reasons: (a) peer 3 already has $1/X$ enhancement layer and it needs to request the other (1-$1/X$) enhancement layer; (b) similar to the affected peers in case 1, both large-window neighbor peers and small-window neighbor peers can support peer 3's resizing. As peer 3 maintains the base layer stream, this switching has no impact on its neighbor peers.

978-1-4244-2324-8/08/$25.00 © 2008 IEEE.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

For a large-window peer requesting both of the two layers, there is no bandwidth wasted, while for a small-window peer, it requests an extra portion $(1/X)$ of the data in the enhancement layer, which cannot benefit it. PFS wastes some bandwidth to improve the performance, but the value of X can be modified to account for the bandwidth limitation. Obviously, BFS is a special case of PFS when $1/X \approx 0$, similarly when $1/X$ equals 100%, PFS becomes QFS.

It is reasonable that we make the small-window peers participate in the distribution of the enhancement layer. First, we only require the small-window peers whose resources are not exhausted to forward the extra data. Second, while the small-window peers and large-window peers dynamically transform to each other, forwarding the extra data has potential benefits: (a) for the case of a large window being resized to a small window, the peer still forwards some enhancement data thus relieving the impact of the resizing on neighbors; (b) for the case of a small window resizing to a large window, the amount of newly requested data at the peers is reduced from 100% to (1-$1/X$) compared with BFS, and thus the resizing time can be reduced. Third, although smart overlay construction algorithms can produce an organized overlay, the two types of peers might occupy random positions in the P2P system because of dynamic window resizing: for example, large-window peers might be surrounded by small-window peers in the topology, which causes content bottleneck - they cannot obtain the media content from their neighbors in spite of sufficient bandwidth between them. PFS eliminates this problem as small-window peers also participate in the enhancement layer distribution.

How much the extra enhancement layer data should be, depends on many aspects: distribution of large/small window peers, resizing frequency, peers' bandwidths, number of neighbors and so on. First, the more the large-window peers that exist in the P2P streaming system, the higher the probability for the existence of large-window neighbors of peers is. So the impact of resizing on both the resizing peer and its neighbors is relieved, and the small-window neighbors just need to forward a small portion of the extra data. Otherwise we should increase the value of $1/X$. Second, if resizing is frequent, forwarding additional enhancement layer data reduces the resizing time. Third, the more neighbors a peer has, the less the burden for each neighbor is, and thus the smaller the value of $1/X$ should be. In real life, the bandwidths for end users vary from several hundred Kbps to several Mbps, and in a P2P streaming system most of the peers connect with dozens of neighbors. In our simulations we set all these parameters close to the real-life values.

V. Simulation Results

A. Simulation Configuration and Metrics

To focus on the dynamic resizing problem in P2P multimedia streaming, in our experiments, we simulate a single channel running on a P2P system consisting of up to 1000 peers with bandwidth distribution as follows: 23% peers with upload bandwidth 1 Mbps and download bandwidth 3 Mbps, 46% peers with 384 Kbps and 1.5 Mbps, and 31% peers with 128 Kbps and 768 Kbps. The average resizing interval is 500 seconds by default. We also run additional simulations in which the bandwidths are more abundant (or more limited) and uniformly distributed among peers. The results of those simulations follow the same trend as presented here, except that when bandwidth is not enough to support QFS, the advantages of PFS are more obvious.

We use a random scheduling algorithm for peers requesting packets; if we use some other algorithms, such as rarest-first [8], the performance may vary, but the trend and relative order among them are expected to be the same.

To simplify the description, below we consider the base layer, and only one enhancement layer. The ratio of stream rates between them is 1:2 (in simulations, they are 100 Kbps and 200 Kbps correspondingly). This setting is similar to MPEG-4 FGS structure [6].

We define the following evaluation metrics that we use in the analysis:

- Average Playback Continuity: We periodically monitor the quality by the average playback continuity [18] - the average ratio of the number of packets received by a peer over the number of packets that should be received, to measure how much the P2P streaming system is affected by resizing under different schemes correspondingly.
- Resizing Delay: We measure playback continuity for each peer resizing from a small window to a large window to see how long the resizing procedure will take to achieve high playback continuity.
- Upload Bandwidth Requirement: We calculate the average required upload bandwidth for a peer in order to measure the total bandwidth requirements for the entire system.

B. Performance Evaluation

We would like to reiterate that BFS and QFS are two special cases of PFS, and we would like to identify some criteria for determining how much the extra enhancement layer data should be in PFS for both high quality and low cost.

Figure 4 shows resizing delay with different values of $1/X$. We can see that as more extra data is forwarded by small-window peers, a shorter delay can be obtained. If $1/X$ equals to 1, resulting in QFS, the playback continuity is not affected by resizing, and the delay is 0. BFS takes almost 5 seconds to achieve high playback continuity (>95%), while even with a small value of $1/X$ such as 1/16, the improvement is still notable.

Figure 5 shows a measure of the average quality of the P2P streaming system under the three schemes. QFS and the PFS with different $1/X$ all have stable and high quality playback continuity, while BFS's performance fluctuates, and is always poorer than the others. There are some sudden performance drops, some more than 20%, in BFS, which lasts for tens of seconds. The reason for this behavior is as follows. Although the structure of the P2P streaming system overlay is mesh-based, there exist some peers close to the source (called

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.

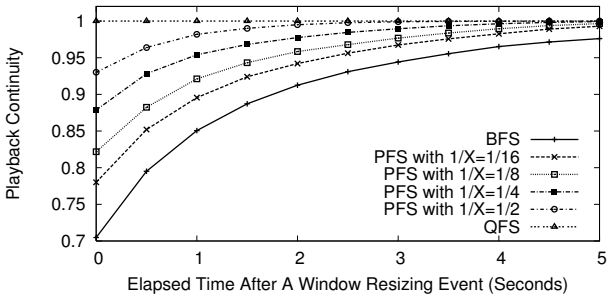

Fig. 4. When a user changes its window size from small to large, the larger the $1/X$, the shorter delay it takes for its playback continuity to reach a desired value.

upstream peers), and peers far away from the source (called downstream peers). In BFS, if upstream peers resize their windows smaller, the downstream peers cannot receive any enhancement layer packets. Those peers need to find new upstream neighbors and the performance drops until the new neighbors are found.

BFS's poor performance is because of low utilization of connections between peers with different window sizes. A connection from a large window peer to a small-window peer cannot contribute to enhancement layer distribution. The effective connections for enhancement layer distribution is much smaller than the total number of connections especially when the average percentage of large peers is low.

Why not construct different overlays for different window size peers or for different layers? First, as resizing is dynamic and stochastic, a peer can randomly resize its window. If it joins one overlay, it needs to find a new one and join it when resizing happens, and this takes a relatively long time and causes stalls. If the peer joins all overlays, it needs to maintain lots of overlay information which increases the system's overhead. Second, there may exist a scenario as follows: there are only a few peers watching a particular channel with a specific window size, and there are not enough peers to construct an efficient and robust overlay for this specific window size.

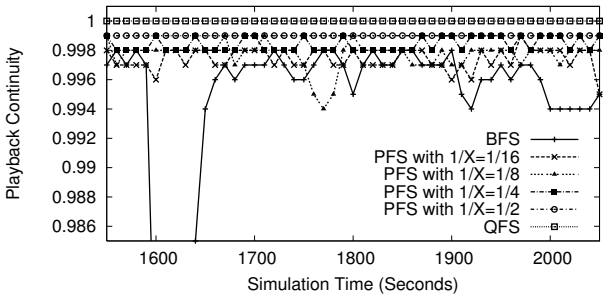

Fig. 5. Small-window peers forwarding $1/X$ of the enhancement layer makes the system's playback continuity high and stable, even with small $1/X$ values.

In Figure 6, we compare the bandwidth requirements of all the schemes. BFS's average upload bandwidth requirement is about 190 Kbps including about 20 Kbps for control messages. With an increase of $1/X$, the bandwidth requirement also increases. The average bandwidth required by QFS is about 325 Kbps. Although QFS outperforms the other two schemes, its cost is the highest as well. Compared with QFS, average bandwidth saved by PFS with different $1/X$ values varies from 120 Kbps (about 40% with $1/X$=1/16) to 65 Kbps (about 20% with $/1/X$=1/2). Combining Figures 4, 5 and 6, we conclude that PFS with a reasonable value of $1/X$ can achieve a short resizing delay, a stable and high overall performance and low bandwidth requirements.

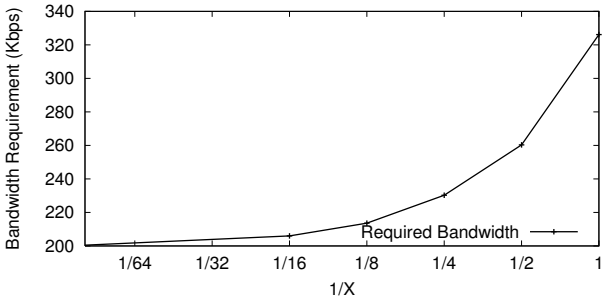

Fig. 6. PFS with different $1/X$ values has different bandwidth requirements.

With $1/X$=1/4 and $1/X$=1/2 the resizing delay (for playback continuity >95%) is about 1 second, and the whole network playback continuity is stable and high (>99%). We adopt the values of $1/X$ of 1/4 and 1/2 to investigate the impact of average window resizing interval and the number of neighbors.

Figures 7 and 8 show that with various average window resizing intervals, PFS can still achieve a short resizing delay (about 1 second) and a high playback continuity (>98%). For the whole network, the longer the average window resizing interval is, the stabler the network is, and consequently the higher the quality that can be achieved. A large value of $1/X$ can relieve resizing impact and the effect is more obvious when resizing activities are frequent. From Figure 7, we can see that frequent resizings shorten the delay, which seems to be counterintuitive. We believe this is because resizing activities make the large window peers and small window peers (with different $1/X$ enhancement layers) more evenly balanced across the overlay, so those peers resizing to a large window can easily collect all the enhancement layer directly from their neighbors. Thus the whole network's performance is harmed by frequent resizings while the local performance may benefit from them.

Figures 9 and 10 show that the performance is closely affected by the number of neighbors. The more neighbors a peer has, the higher its probability to collect all of the enhancement layer from its neighbors when it resizes from a small window to a large window. As the number of neighbors for a peer increases, the probability for a peer to find other peers to support it from its neighbors also increases when its upstream peer resizes from a large window to a small window. When the number of neighbors is already relatively large, increasing the number of neighbors further does not improve

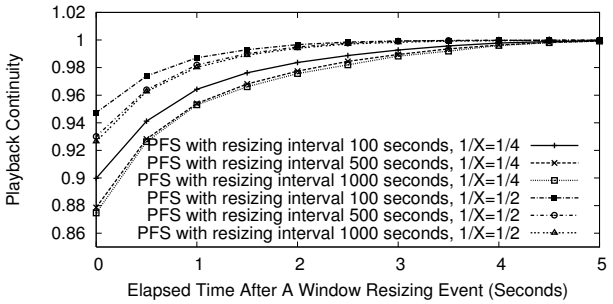

Fig. 7. When a user changes its window size from small to large, it quickly achieves high playback continuity using PFS with $1/X$=1/4 and $1/X$=1/2 under various average window resizing intervals.

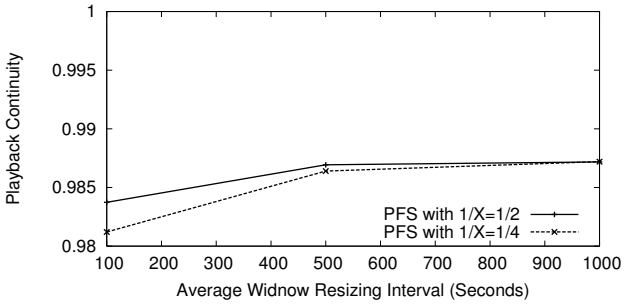

Fig. 8. The longer the average window resizing interval is, the higher average playback continuity is.

the performance much. The figures also show that with less neighbors, a larger value of $1/X$ is needed.

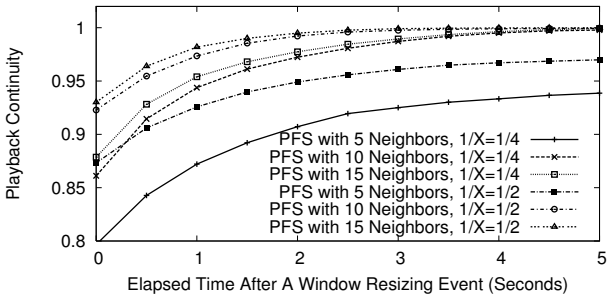

Fig. 9. When a user changes its window size from small to large, the more neighbors it has, the shorter delay it takes for its playback continuity to reach a desired value.

VI. Conclusion and Future Work

In this paper, we investigated how to achieve the desired streaming quality in case of dynamic window resizing in live P2P streaming systems with relatively low bandwidth requirement, and proposed PFS based on layered coding. Simulation results show a scheme PFS relieves the impact of dynamic window resizing with low bandwidth requirement.

We proposed several guidelines to determine the value of $1/X$ under various scenarios, but have not provided a formula to calculate it. In future work we propose to conduct a theoretical analysis, expand the scheme to several layers for multi-channel applications and study an incentive-based mechanism for determining the value of $1/X$.

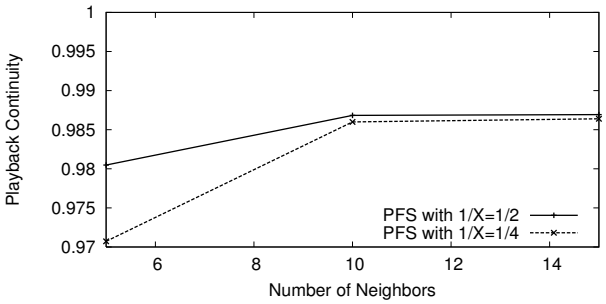

Fig. 10. The more neighbors a peer has, the higher average playback continuity can be achieved.

References

[1] J. Chakareski, S. Han, and B. Girod. Layered coding vs. multiple descriptions for video streaming over multiple paths. In *ACM Multimedia 2003*, Berkeley, California, November 2003.

[2] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proceedings of the 13th ACM NOSSDAV*, Monterey, CA, June 2003.

[3] M. Ghanbari. Two-layer coding of video signals for VBR networks. *IEEE Journal on Selected Areas in Communications*, 7(5):771–781, June 1989.

[4] V. K. Goyal. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, 18(5):74–93, September 2001.

[5] H. Jagadish, B. C. Ooi, K. L. Tan, Q. H. Vu, and R. Zhang. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *Proceedings of International Conference on Management of Data*, Chicago, Illinois, June 2006.

[6] T. Kim and M. H. Ammar. Optimal quality adaptation for scalable encoded video. *IEEE Journal on Selected Areas in Communications*, 23(2):344 – 356, 2005.

[7] Y. Lee, J. Kim, Y. Altunbasak, and R. M. Mersereau. Performance comparisons of layered and multiple description coded video streaming over error-prone networks. In *Proceedings of International Conference on Communications*, Anchorage, Alaska, May 2003.

[8] A. Legout, G. U. Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proceedings of Internet Measurement Conference*, Rio de Janeiro, Brazil, October 2006.

[9] J. Li. Peer-to-peer multimedia applications. In *Proceedings of the 14th annual ACM International Conference on Multimedia*, Santa Barbara, CA, October 2006.

[10] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang. Using layered video to provide incentives in P2P live streaming. In *Proceedings of IPTV*, Kyoto, Japan, August 2007.

[11] N. Magharei, Y. Guo, and R. Rejaie. Issues in offering live P2P streaming service to residential users. In *Proceedings of IEEE Consumer Communications and Network Conference*, Las Vegas, January 2007.

[12] N. Magharei and R. Rejaie. PRIME: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[13] V. Padmanabhan, H. Wang, and P. Chou. Resilient peer-to-peer streaming. In *Proceedings of ICNP*, Atlanta, Georgia, November 2003.

[14] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *In The Fourth International Workshop on Peer-to-Peer Systems*, Ithaca, New York, February 2005.

[15] PPLive. http://www.pplive.com.

[16] PPStream. http://www.ppstream.com.

[17] M. Zhang, Y. Xiong, Q. Zhang, and S. Yang. Optimizing the throughput of data-driven peer-to-peer streaming. *IEEE Transactions on Parallel and Distributed System, to appear*, 2008.

[18] X. Zhang, J. Liu, B. Li, and T. Yum. DONet/CoolStreaming: A data-driven overlay network for live media streaming. In *Proceedings of IEEE INFOCOM*, Miami, Florida, March 2005.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2008 proceedings.