Quality- and Context-aware Neighbor Selection for Layered Peer-to-Peer Streaming

Anh Tuan Nguyen [†], Baochun Li [††], and Frank Eliassen [†]
[†] *Department of Informatics, University of Oslo, Oslo, Norway*
{anhtn,frank}@ifi.uio.no
[††] *Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada*
bli@eecg.toronto.edu

***Abstract*—Layered streaming is being considered as the most promising approach to adapt to bandwidth variations and heterogeneous end users in streaming applications. The goal of a layered streaming protocol is not only to optimize the average playback skip rate as in single-layer streaming, but also to maximize possible quality level (quality satisfaction) based on the available bandwidth capacity at the end user. In unstructured layered peer-to-peer streaming, however, achieving high quality satisfaction is challenging due to content and bandwidth bottlenecks. With experiments, in this paper, we demonstrate the importance and identify unique challenges of neighbor selection to the system performance in terms of the average skip rate and quality satisfaction. Then, we propose a new neighbor selection technique that can offer good performance while keeping the scalability of the mesh overlay under network fluctuations. The core of the technique is a *preemption rule* that allows a higher capacity peer to replace a lower capacity peer to be a neighbor of another peer with a certain probability. This preemption rule gears high capacity peers to *good* locations in the overlay to maximize the use of their bandwidth capacity and available layers. Simulation results demonstrate the efficiency of the method.**

I. Introduction

Layered streaming has been an attractive research topic for years because it can adapt to bandwidth variations and heterogeneous end users. The adaptability is very much required when video streams are transferred across best-effort IP networks. Compared to traditional streaming systems, layered streaming enables high capacity users to receive high quality video, while low capacity peers still enjoy an acceptable quality level. In addition, when bandwidth drops, instead of suffering playback skips, users are able to perceive smooth playback with reduced quality. With the reality of peer-to-peer (P2P) streaming, the use of layered coding in P2P streaming is beneficial to provide adaptive streaming to a large number of users with low cost servers. However, layered P2P streaming also poses unique challenges, of which one of the most difficult problems is the overlay construction that needs to be designed carefully to mitigate content and bandwidth bottlenecks.

P2P overlays can be structured or unstructured (mesh-based). Structured overlays ease the data delivery. However, since they are highly affected by peer dynamics, an additional protocol is usually run to restore/reshape the overlay structure. On the other hand, unstructured overlays make the system more robust to network fluctuations without the need of a global mechanism to maintain the overlay. Therefore, unstructured overlays are more suitable for P2P streaming in dynamic environments, e.g. the Internet [1].

In an unstructured P2P overlay, connectivity between peers is created by the neighbor selection method run at each peer. Such neighbor selection forms a partial view of the system from which a subset of peers is chosen to exchange video data. Therefore, it is the neighbor selection that determines the perceived quality at each peer.

Owing to its important role, there have been many studies on neighbor selection in single-layer P2P streaming. However, they are insufficient when applied to layered P2P streaming because of the following reason. In single-layer P2P streaming, peers receive the same video stream. Meanwhile, in layered P2P streaming, the video stream is encoded into quality layers, and peers aim to receive the maximum number of layers according to their available bandwidth capacity. Therefore, while the average playback skip rate is the main performance metric in single-layer P2P streaming, the ratio of the experienced quality level and the expected quality level determined by the bandwidth capacity, called *quality satisfaction*, is also an important metric in layered P2P streaming. Some early studies in layered unstructured P2P streaming focus on peer coordination to achieve high quality satisfaction by determining which peers to communicate for data exchange, but the neighbor selection has not received much attention. We believe that *quality-* and *context-aware* neighbor selection will boost the layered streaming protocol.

In this paper, with experiments on our new adaptive streaming protocol, named *Chameleon* [2], we find out factors that impact the quality satisfaction of each peer. In particular, peers are classified into classes based on their bandwidth capacity in which peers of class C_i (class identifier) have higher bandwidth capacity than peers of class C_j, if $i > j$. The quality level peers in a certain class C perceive is impacted by (1) the join order of peers of different peer classes, and (2) the percentage in population of class C in the system. We then propose a new neighbor selection technique with a sole objective of providing peers high quality satisfaction regardless of their join time and the population of their class in the system. The core of the technique is a *preemption rule* that allows a higher capacity peer to replace a lower capacity peer to be a neighbor of another peer with a certain probability. This preemption rule gears high capacity peers to *good* locations in the overlay to maximize the use of their bandwidth capacity and available layers. For example, high capacity peers stay closer to the server than low capacity peers, and high capacity peers have more privilege to be neighbors of other high capacity peers than low capacity peers.

Simulation results demonstrate that our simple but effective approach is able to achieve high quality satisfaction for each peer regardless of the two aforementioned factors.

The rest of this paper is organized as follows. Section II discusses related work. Section III identifies the problems of neighbor selection with different peer join patterns and percentages of peer classes in the peer population. The proposed neighbor selection method is described in section IV. Simulation results are discussed in Section V. Finally, Section VI concludes the paper.

II. Related Work

There has been a substantial amount of research attention on overlay construction in single-layer P2P streaming. Some studies focus on structured overlays, e.g., tree-based overlays [3]–[5], while others spend efforts on unstructured overlays [6]–[8]. Magharei *et. al.* [1] and Seibert [9] *et. al.* present comparisons of P2P streaming approaches, in which it is demonstrated that mesh-based approaches consistently exhibit a superior performance over tree-based approaches in dynamic environments, while, in stable environments, tree-based systems are better in terms of delivery time. To take advantage of both approaches, efforts have been spent on hybrid overlays, such as [10].

In layered P2P streaming, [11]–[13] present multiple-tree based systems using multiple description coding to provide differentiated service. The general idea is that each video description is delivered in one tree, and peers can receive more than one description by being a node in more than one tree. However, early work in unstructured layered P2P streaming focuses on peer coordination, rather than overlay construction, to maximize quality level each peer perceives [14], [15].

More related to our work, Zhao *et al.* propose LION [16], a layered overlay multicast system. LION progressively organizes peers into layered meshes. Within each mesh, the delivery of one quality layer is carried out with using network coding. Each peer can subscribe to a proper number of meshes to maximize its throughput by fully utilizing its available bandwidth. Our approach is similar to theirs in the way that we also explore mesh-based overlays and network coding in layered streaming, but we use only one mesh for the delivery of all layers. In addition, although each video layer is delivered in a mesh which is unstructured, the whole overlay structure of LION is quite well-organized and maintained by a distributed heuristic algorithm, derived from a complicated optimization problem in mathematical programming. Therefore, LION is aimed to support small-scale application scenarios in stable environments. On the other hand, our method is inherent distributed and is geared towards the goal of building up an adaptive and scalable P2P streaming system on a dynamic overlay. The simplicity and efficiency are two most important criteria in our design.

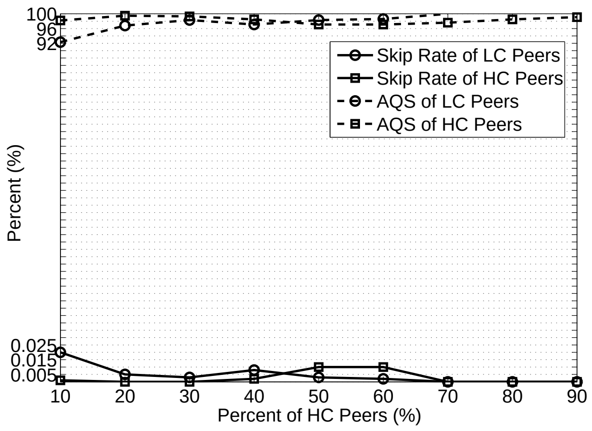
Another related work is OCals proposed by Xiao *et. al.* [17]. Ocals constructs the overlay in two stages. The first stage is to probe existing nodes to find a certain number of logical partners, which are interested in the same set of layers. In the second stage, it will select neighbors for each layer based on RTT (Round Trip Time). Similar to OCals, we focus on neighbor selection and agree that peers with similar interests in terms of the number of quality layers should connect to each other. However, there are two main differences between our work and OCals. Firstly, in our approach, a peer i chooses neighbors based on the current quality level candidates are perceiving, not based on logical partners as in Ocals, which is similar to class-based selection mentioned in [2] when logical partners are classified into the same class. We have shown in [2] that quality-based selection is better than class-based selection in terms of both important performance metrics, the average playback skip rate and quality satisfaction. Secondly, we choose a neighbor j with a certain probability, while Ocals makes a selection based on comparisons with exact values (medium and minimum RTTs). As widely used in other studies, e.g., [18], [19], we believe that adding some degree of randomness (through probability) to neighbor selection creates a more robust overlay with respect to network fluctuations.

III. Problem Identification

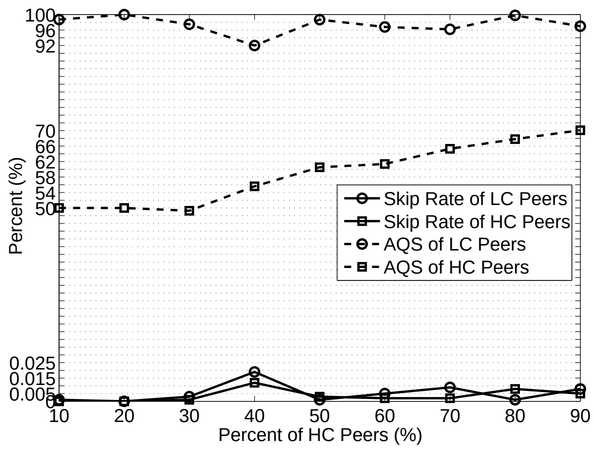
We consider a typical P2P streaming session with a number of dedicated streaming servers, and a large number of peers. Peers participate in and depart from a session in unpredictable ways, and they are heterogeneous with different bandwidth capacities. Peers can be classified into classes based on their bandwidth capacity. A layered coding technique, e.g. SVC [20], is used to encode raw video data into quality layers. Peers are organized in an unstructured overlay, i.e., there are no global mechanisms to build and maintain the overlay. We assume that every peer is willing to contribute its bandwidth to upload data to other peers, i.e., no selfish or fraud peers in the system.

Although we focus on neighbor selection, we need a complete streaming protocol to evaluate the proposed method. We use Chameleon, our adaptive streaming protocol, which has been shown to be able to achieve good performance in terms of the average playback skip rate and quality satisfaction [2]. We also use the quality-based neighbor selection in Chameleon as a baseline method in this study, which is summarized as follows. When a peer joins the system, or when it needs to update the neighbor list for better quality, it creates neighborships with other peers. A list of available peers can be provided by a rendezvous peer or by exchanging membership information, e.g. using SCAMP [21]. Each peer calculates the average quality level it has perceived so far. When a peer selects a neighbor, it will choose the peer(s) whose average quality level is closest to its class identifier within a range τ. If there are more peers than needed, a subset is selected based on the peer class in the following order: peers in the same class, peers in higher classes, and peers in lower classes.

With Chameleon, we have generated the join time and the class of each peer randomly, and we have observed notable performance differences between experiments with significantly different patterns of peer join. Therefore, to reveal the effect of the join order and the population percentage of each peer class to the system performance, we consider two extreme cases in the following experiments. Without any loss of generality, we use the JSVM Software [22] to generate a two-hour video sequence with two quality levels, and we classify peers into two classes: high capacity (HC) and low capacity (LC). The download and upload capacity are set so that HC peers are able to receive two quality levels (the full quality) while LC peers are only able to receive one quality level (the base level). In particular, we set the download and upload capacity of peers to 6-10% and 4-8% higher than the stream rate of the quality level corresponding to each peer class. There are no super peers in the system. We use only one server, which can serve 8-10% of the total number of peers in the system. We evaluate the performance of Chameleon with the quality-based neighbor selection method in Case A: all HC peers join the session before LC peers, and Case B: all HC peers join after LC peers. In each case, the number of HC peers is set to 10, 20, ..., and 90% of the peer population.



(a) Case A



(b) Case B

Fig. 1. Performance of Chameleon in the two cases.

Figure 1 shows the performance of Chameleon in Case A and Case B. In general, the average skip rates are very low in both cases, but the average quality satisfaction (AQS) is very different. In Figure 1(a), when HC peers join the system first, the average quality satisfaction for both classes is very high ($> 92\%$) regardless of the number of HC peers in the system, which means each peer can perceive 92% of its best possible quality level according to its bandwidth capacity. On the other hand, in Figure 1(b), when LC peers join first, the average quality satisfaction of HC peers is low and increases from 50% to 70% when the percentage of HC peers increases. The reason is that, in Case B, LC peers join first, connect and stay close to the server in terms of the number of hops from the server. Since the number of connections the server can create is limited, when HC peers join, they may be not able to connect directly to the server and content bottlenecks occur. For example, when there is only 10% HC peers in the system, all HC peers receive only the base layer. Consequently, the quality satisfaction is 50% (because they are expected to be able to receive two quality levels). When the population percentage of HC peers increases, the chance of connecting to the server increases, and the top layer can be delivered to some HC peers. In the other case, if HC peers join the session first and connect to the server, they can receive the top quality layer from the server and deliver it to other HC peers. In addition, since they also have the base layer, LC peers are served well.

From the above experiment, we observe that the join order and the percentage in population of different peer classes may not affect the performance of single-layer P2P streaming systems much, as the skip rates are low in the two cases, but they do impact the average quality satisfaction of different peer classes in layered P2P streaming systems. The question here is: *How to provide high average quality satisfaction for different peer classes regardless of their join time and population*? We answer this question in the next section with our scalable and effective neighbor selection method.

IV. A Quality- and Context-Aware Neighbor Selection Method

The difficulty in designing a neighbor selection method is that each peer only knows information of a certain number (not all) of peers in the system. Although it is possible to update information about the population of each peer class in the system by tracking join requests at rendezvous peers, it causes traffic overhead to transmit the information, which changes frequently when peers join and leave. To keep the system scalable, our proposed method is based only on local information of candidates to choose neighbors.

As in single-layer streaming, it is reasonable that high capacity peers should have higher priority than low capacity peers in being located at *good* positions in the overlay, e.g., close to the server or other high capacity peers because when they can receive more, they will contribute more to other peers in terms of bandwidth and layers. Based on this fact and taking the effect of the peer join order into account, we use a *preemption rule* as follows: when a peer P in class C_i wants to connect to a peer Q which has reached its maximum number of neighbors defined by the system, if one neighbor K of Q belongs to class C_j, $C_j < C_i$, then P can replace K to be a neighbor of Q. However, if the rule is applied strictly everywhere in the overlay, peers in the lowest class (the lowest bandwidth capacity) can only connect to each other and create clusters of low quality peers, which will suffer high playback skip rates. Therefore, a peer P in class C_i should only be able to replace another peer K in class C_j, $C_i > C_j$, with a probability P_preemp, P_preemp is higher when K is *closer* to the server. This rule guides high capacity peers closer to the server even if they join the system after low capacity peers, and low capacity peers further from the server even if they join the system before high capacity peers. We calculate the distance between a peer P and the server by the minimum number of peers (hops) between P and the server through neighborship with the following algorithm:

- ▷ The distance from the server to itself is 0.
- ▷ The distance from a peer P to the server is calculated by the minimum distance of its neighbors to the server $+1$.

$$D_P = \min_{i \in NL_P}(D_i) + 1$$

  in which D_i is the distance of peer i to the server, and NL_i is the neighbor list of peer i.
- ▷ The distance of a peer is updated when its neighborship changes, e.g., a neighbor is added or deleted.

It is noted from the above algorithm that it is not required to update the distance of each peer in a timely manner, e.g., when the distance from a peer to the server changes, the distance from its neighbors to the server may also be changed but is not updated. The update algorithm is only invoked at a peer when its neighborship changes. The reason for this *relative* calculation is that the timely update requires message exchanges, which cause traffic overhead, between peers to inform their distance has changed. In addition, as being demonstrated later, the relative distance is good enough to point out the *vicinity* of the peer location in the overlay. When each peer maintains its distance to the server, the preemption probability P_preemp to replace a peer K is calculated by the following formula.

$$P\_preemp = \frac{1}{1 + \alpha(D_K - 1)} * 100$$

in which α is a tunable parameter. From this formula, we can see that if low capacity peers connect directly or stay close to the server (because they join the session first), they are likely replaced by high capacity peers. For example, if a low capacity peer connects directly to the server, i.e. its distance is 1, then the $P\_preemp = 100\%$, so it will be replaced by a high capacity peer. However, if low capacity peers are far from the server, they can keep their connections to high quality peers to maintain their quality, as the preemption probability is low. The value of α determines how P_preemp reduces on the way far from the server, e.g., if $\alpha = 1$, P_preemp for $D = 1, 2, 3, 4, ...$ is 100%, 50%, 33.33%, 25%, ... respectively. Currently, we choose α by experiments. However, how to choose a good α in general cases is an important issue, and we leave it as our future work.

We now are ready to present the complete neighbor selection method. A peer will create neighborships when it joins the system or when it wants to improve the video quality, e.g., a neighbor leaves the system, or the current quality level drops below a threshold for a period of time. When a peer P needs one or more neighbors, it:

- ▷ contacts a rendezvous peer with necessary information such as its estimated bandwidth capacity. The rendezvous peer will return a list of available peers in the system and the class identifier P belongs to.
- ▷ sends requests containing its class identifier to all candidates, who are selected by the quality-based neighbor selection method in [2].
- ▷ on receiving notifications from the candidates, creates connections and stores neighbor information to the neighbor list, or waits for a period of time and tries again.

Algorithm 1 Neighbor Selection - Request

```
AL: the peers list returned by the rendezvous peer.
CL: the candidate list.
NL: the neighbor list.
CL ← QualityBasedSelect(AL);
SendRequest(CL);
while (active) do
  if (accept(Q)) then
    Insert(Q, NL);
  end if
end while
if (too_few_neighbors) then
  Wait();
  Request();
end if
```

For each candidate, on receiving a request:

- ▷ if the current number of neighbors is below its maximum number of neighbors, accepts the request.
- ▷ otherwise, selects the neighbor whose class identifier is lowest, calculates its P_preemp and decides to accept the request with the probability P_preemp.

Algorithm 2 Neighbor Selection - Respond

```
N: the current number of neighbors.
MAX_N: the maximum number of neighbors.
NL: the neighbor list.
ReceiveRequest(P);
if (N < MAX_N) then
  SendReply(P, ACCEPT);
  Insert(P, NL);
else
  R ← SelectLowestCapacityPeer(NL);
  if (P.class > R.class) then
    P_preemp ← $1/(1 + \alpha *$ (R.distance $- 1$));
    if (Rand() < P_preemp) then
      SendReply(P, ACCEPT);
      Insert(P, NL);
    else
      SendReply(P, REJECT);
    end if
  end if
end if
```

V. Simulation Results

In this section, we revise the two extreme cases in Section III with the proposed selection method, and demonstrates its efficiency in more general cases with different network sizes to check the system scalability. Finally, we consider features of the topology formed by the method. We implement the proposed neighbor selection method in Chameleon and evaluate its performance in terms of the average playback skip rate and the average quality satisfaction. The bandwidth settings are the same with the settings in Section III.

A. The proposed method offers adaptability and scalability

Figure 2 shows the performance of Chameleon in Case B (the graph for Case A is similar) when LC peers join the session before HC peers. It is clearly observed that the quality satisfaction of HC peers is significantly improved compared to Figure 1(b). Thanks to the preemption rule, HC peers can gradually be located in good positions. This not only enables HC peers achieving high quality video, but also minimizes the skip rate of LC peers.

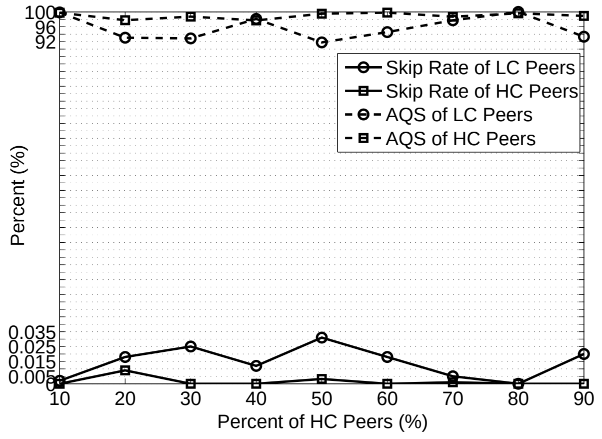


Fig. 2. The system performance with the proposed protocol in Case B

In practice, peers can join and leave the system at any time. Therefore, we now come back to a more general case. We generate another video sequence with four quality layers and use four classes of peers corresponding to the four quality layers. The join time and class identifier of each peer is generated randomly, and the peer life time is generated by the Weibull distribution – Weibull(k, 2) – as shown in [23] that the peer session lengths are fit by the Weibull distribution. Figure 3 shows the average skip rate and quality satisfaction of each peer class. It demonstrates that the neighbor selection method helps to achieve best possible quality for each peer class in the system, while keeping the system scalable under peer dynamics.

B. Features of the topology

An interesting question here is that: *what does the topology actually look like under the neighbor selection method?* To answer the question, for every peer in class C, we calculate the percentage of its neighbors which belong to class C_k, $k = 1, 2, 3, 4$. In this experiment, the network size is 700, every peer has an average of 55 neighbors. The population percentage of class 1, 2, 3, and 4 in the system is 25.71%, 24.29%, 16.99%, and 33.01%, respectively. Table I shows the neighboring relationships between the peer classes. The value at element (i, j), $T(i, j)$, is the average percentage of peers of class j in the neighbor list of peers of class i.

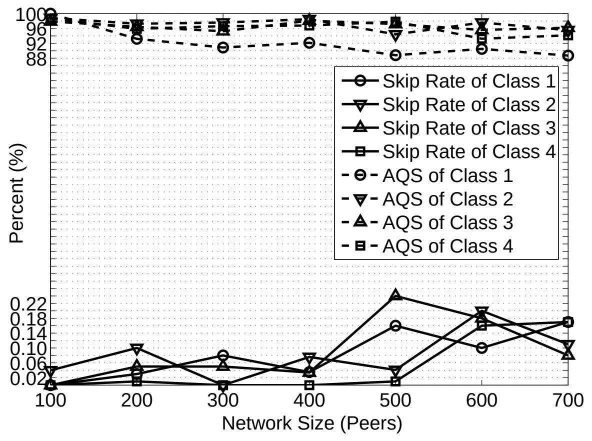


Fig. 3. Performance of Chameleon with different network sizes

TABLE I
TOPOLOGY

Peer class	1	2	3	4	Avg. Distance
1	**73.51**	22.42	3.96	0.11	4.14
2	22.22	**54.25**	16.99	6.54	3.47
3	7.07	29.30	**42.42**	21.21	2.73
4	3.29	11.93	12.35	**72.43**	1.92

As shown in Table I, the neighbor selection method creates clusters of peers that belong to the same class: $T(i, i) \geq T(i, j), \forall i, j$, i.e., peers of the same class tend to connect to each other. In addition, we also calculate the average distance of peers of each class to the server. The result is presented in the right most column of Table I, which shows that the higher the class identifier is, the closer to the server the class is. In summary, with the proposed neighbor selection method, peers are grouped into clusters based on their peer class, and clusters of higher capacity peers stay closer to the server than those of lower capacity peers. This can be considered as an *emergent property* of the method, because each peer selects its neighbors with partial knowledge about the network.

VI. CONCLUSION

In this paper, we point out that, different from single-layer P2P streaming, the join order of peers and the population percentage of different peer classes in the system may impact the system performance in layered P2P streaming. We then propose a new neighbor selection method that uses the perceived quality level and location information of candidates to choose neighbors. Simulation results demonstrate the adaptability and scalability of the proposed method. Supported by our results in [2], we can indirectly infer that Chameleon with this new neighbor selection method offers even better performance when compared with the related work. We are following two directions for future work. Firstly, we are interested in the *'evolution'* of the overlay over time. In particular, when many peers with different peer classes join the system or when a severe network congestion occurs, when will each peer (again) receive its best possible quality? Secondly, we are studying combinations of the preemption rule with other network metrics, e.g. RTT, to achieve a more efficient and practical method. Then, we are able to evaluate Chameleon at network level to demonstrate is performance in real world settings.

References

[1] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or multiple-tree: A comparative study of live p2p streaming approaches,” in *Proc. of IEEE INFOCOM*, May 2007, pp. 1424–1432.

[2] A. T. Nguyen, B. Li, and F. Eliassen, “Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding,” *To be appeared in Proc. of IEEE INFOCOM 2010*, July 2009.

[3] Y. Chu, S. Rao, S. Seshan, and H. Zhang, “Enabling conferencing applications on the internet using an overlay muilticast architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 55–67, 2001.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable Application Layer Multicast,” in *Proc. of SIGCOMM*. New York, NY, USA: ACM, 2002, pp. 205–217.

[5] D. A. Tran, K. A. Hua, and T. Do, “ZIGZAG: an efficient peer-to-peer scheme for media streaming,” in *Proc. of INFOCOM*, vol. 2, 3 April 2003, pp. 1283–1292.

[6] N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming,” in *Proc. of IEEE INFOCOM*, May 2007, pp. 1415–1423.

[7] F. Pianese, D. Perino, J. Keller, and E. W. Biersack, “PULSE: An Adaptive, Incentive-Based, Unstructured P2P Live Streaming System,” *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1645–1660, Dec. 2007.

[8] X. Zhang, J. Liu, B. Li, and T.-S. Yum, “CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming,” in *Proc. of IEEE INFOCOM*, vol. 3, March 2005, pp. 2102–2111.

[9] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru, “Experimental comparison of peer-to-peer streaming overlays: An application perspective,” in *LCN 2008*, Oct. 2008, pp. 20–27.

[10] F. Wang, Y. Xiong, and J. Liu, “mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast,” in *Proc. of ICDCS*, June 2007, pp. 49–49.

[11] J. D. Mol, D. H. P. Epema, and H. J. Sips, “The Orchard Algorithm: Building Multicast Trees for P2P Video Multicasting Without Free-Riding,” *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1593–1604, Dec. 2007.

[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: high-bandwidth multicast in cooperative environments,” in *SOSP ’03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2003, pp. 298–313.

[13] N. P. Venkata, H. J. Wang, and P. A. Chou, “Resilient Peer-to-Peer Streaming,” in *ICNP*, 2003, pp. 16–27.

[14] R. Rejaie and A. Ortega, “PALS: peer-to-peer adaptive layered streaming,” in *Proc. of NOSSDAV*, Monterey, CA, USA, jun 2003, pp. 153–161.

[15] Y. Cui and K. Nahrstedt, “Layered peer-to-peer streaming,” in *Proc. of NOSSDAV*, Monterey, CA, USA, jun 2003, pp. 162–171.

[16] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, “LION: Layered Overlay Multicast With Network Coding,” *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1021–1032, October 2006.

[17] X. Xiao, Y. Shi, B. Zhang, and Y. Gao, “OCals: A Novel Overlay Construction Approach for Layered Streaming,” in *Proc. of ICC*, May 2008, pp. 1807–1812.

[18] D. Hales, S. Arteconi, and O. Babaoglu, “SLACER: Randomness to Cooperation in Peer-to-Peer Networks,” in *Proc. of Collaborative Computing Networking, Applications and Worksharing*, 2005.

[19] F. E. Bustamante and Y. Qiao, “Friendships that Last: Peer Lifespan and its Role in P2P Protocols,” in *Web Content Caching and Distribution*. Springer Netherlands, 2007, pp. 233–246.

[20] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[21] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, “Peer-to-peer membership management for gossip-based protocols,” *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 139–149, Feb. 2003.

[22] ITU-T and I. JTC1. (2008) JSVM Software version JSVM 9.17. [Online]. Available: http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm

[23] D. Stutzbach and R. Rejaie, “Understanding churn in peer-to-peer networks,” in *Proc. of ACM SIGCOMM IMC*. New York, NY, USA: ACM, 2006, pp. 189–202.