This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

A Neighboring Strategy for ISP-Friendly Peer-to-Peer Video Live Streaming

Xiangyang Zhang
School of Computing, Queen's University, Canada
Email: xiang@cs.queensu.ca

Hossam Hassanein
School of Computing, Queen's University, Canada
Dept. of Computer Science, King Saud University, KSA
Email: hossam@cs.queensu.ca

***Abstract*—Push-pull hybrid schemes for peer-to-peer (P2P) video live streaming applications achieve a short playback delay and are robust in the presence of peer churn. Most hybrid schemes construct an overlay randomly, which causes unnecessary traffic on the Internet. Simply applying the neighbor-with-nearby-peers strategy can localize the traffic but results in a large tree height, more lost chunks, and increased playback delay. In this paper, we propose a new neighboring strategy to construct a hierarchical overlay that has a low average edge cost and helps to build a short and robust tree. Using a heuristic degree-bounded shortest path tree algorithm and an efficient pull mechanism to recover late chunks, the hierarchical overlay achieves performance close to the random overlay but has only $\frac{1}{3}$ of the network cost; 98% of peers receive all the chunk with playback delays well within the acceptable range.**

I. Introduction

Peer-to-peer (P2P) video streaming applications generate an enormous amount of traffic on the Internet, which causes Internet Service Providers (ISPs) significant expenditure and threatens the quality of other Internet services. P2P video live streaming schemes include tree-based push schemes, swarm-based pull schemes, and push-pull hybrid schemes [1]. Push schemes build a multicast tree, usually with the aim of optimizing a network cost function, to push packets from the video server to each peer. In pull schemes, the video is split into chunks of fixed size; peers exchange buffer maps describing which chunks they have and pull missing chunks from one another. Push schemes are vulnerable to peer churn while pull schemes have a large playback delay. Push-pull hybrid schemes combine tree-building and swarming technique together, achieving a short playback delay while being robust in the presence of peer churn.

P2P systems usually construct a *random overlay* (i.e., peers choose neighbors randomly), which has many favourable properties but causes enormous unnecessary traffic. Several studies on P2P file sharing applications [2]–[5] propose that peers only neighbor with nearby peers to localize traffic, and [3], [4] report reduced file downloading time. However, unlike file sharing, P2P video live streaming has stringent delay requirements. In a recent study [6], we show that using the neighbor-with-nearby-peers strategy can result in an overlay which is prone to partitioning, a multicast tree with a large height, and more chunk losses in the presence of peer churn.

In this paper, we propose a neighboring strategy to construct a *hierarchical overlay* that has low-cost edges and helps to build a short and robust tree. Our scheme allows flexible definition of network cost, addressing ISPs' need to reduce both inter- and intra-autonomous system (AS) traffic. Most edges on the hierarchical overlay are between nearby peers, but a small fraction of edges are rewired to remote peers in a way that the overlay exhibits a hierarchical structure. With a heuristic degree-bounded shortest path tree (DBSPT) algorithm and an efficient pull mechanism to recover late chunks, the hierarchical overlay achieves tree height and chunk delivery rate close to the random overlay but has only $\frac{1}{3}$ of the network cost; 98% of peers receive all the chunk with an average playback delay of 5.6 seconds.

The remainder of this paper is organized as follows. Section II introduces related work. Section III presents the overlay construction scheme. Section IV briefs tree construction and pull mechanism. Section V evaluates the performance of our proposed scheme. Section VI concludes this paper.

II. Related Work

In hybrid P2P video live streaming schemes, peers first construct an overlay then build multicast trees and pull missing chunks on the overlay. In most schemes, such as [7], [8], peers construct a random overlay for its high connectivity, small diameter, and other favorable properties. In [9], a new peer selects some neighbors randomly and some by round trip time (RTT) from its membership table. Because the membership table is small compared with the population, the probability of nearby peers being selected is small. In [10], a new peer selects neighbors, also from its membership table, according to a function of neighbor's residual bandwidth and delay to the source. In [11], peers neighbor with nearby peers but also choose a small number of neighbors randomly to form a "small-world" overlay. The multicast tree construction can be *data-driven* or *network-driven*. The data-driven approach [7]–[10] starts from pull schemes. In addition to pulling missing chunks, peers also subscribe to neighbors, according to the chunks they have and exchange history, to pull future chunks. The network-driven approach [11] starts from push schemes. Peers build a multicast tree to optimize a network cost function; the pull mechanism is used to re-transmit late chunks and has no impact on the tree building.

An ISP-supported service is proposed in [2], [3], which helps P2P applications to select nearby neighbors. In [4], peers

978-1-61284-233-2/11/$26.00 ©2011 IEEE

use a content distribution network (CDN)'s redirecting information to find nearby peers. In [5], peers measure the latency to a set of landmark hosts to infer their network positions. This paper focuses on the neighboring strategy rather than network positioning techniques; it uses existing positioning schemes. The performance of the neighbor-with-nearby-peers strategy in P2P file sharing applications is investigated in [3], [4]; both report reduced AS hops and file downloading time. Network utilization is unaddressed in most P2P video live streaming schemes [7]–[9]. We investigate whether the strategy is applicable in the context of P2P video live streaming in [6]. Results show that although it achieves a low cost, the strategy results in a large tree height, reduced robustness against peer churn, reduced reliability against network element errors, and higher probability of partitioning.

III. Overlay Construction

When peers only neighbor with nearby peers, the overlay has the lowest edge cost but may be partitioned if some peers depart. Isolated cliques need several seconds or more to reconnect. Temporary partitioning is tolerable for file sharing but greatly degrades the service quality in video streaming systems. A common technique to reduce partitioning, while maintaining a low edge cost, is to rewire a small fraction of edges to random peers to form a "small-world" graph [12]. Although both small-world and random graphs asymptotically have a diameter of $O(\log N)$, the actual difference is significant. Small world graphs also have high clustering because most edges are between nearby nodes. Both properties have negative impacts on video live streaming systems.

A. Desired Overlay Properties

We desire a short, low cost multicast tree that is robust against peer churn. A large tree height has several drawbacks for a P2P video live streaming system, even if most hops are within the same AS. First, the system becomes less reliable because a path with more hops has more network elements and each network element has an error rate (e.g., links have transmission errors, routers drop packets when congestion occurs, and multi-tasking hosts may temporarily block P2P clients). Second, the system becomes less robust. Departure of a peer causes chunk losses at its descendants. The probability that a peer has a departing ancestor is proportional to its hop count to the video server. Third, the playback delay becomes larger. In our recent work [6], we show that in data-driven hybrid schemes, peer churn causes multicast trees to grow to a large height and high clustering makes the situation more severe. Therefore, we propose the network-driven approach, and design the overlay in such a way that it has a low edge cost and, together with the DBSPT algorithm, reduces the tree height and mitigates the negative impact of high clustering.

Without loss of generality, we assume peers are distributed on a 2-dimensional plane and distances on the plane represent costs. For convenience, we say an overlay edge is *short* or *long* depending on its cost. We equally divide the range (u_{min}, u_{max}) of peers' upload bandwidth into l segments; peers are assigned into layers depending which segment their bandwidths fall into. Naturally, peers of each layer are evenly distributed on the plane. We first add short edges between nearby peers then rewire a small fraction of edges to form a hierarchical overlay as shown in Fig. 1. We design the overlay to have and retain the following properties in the presence of peer churn. First, high-bandwidth peers are evenly distributed on the overlay, i.e., given a layer i and hop count j, the number of layer i peers in the clique within j hops to any peer x are similar. This property is necessary to utilize peers' upload bandwidth. Second, long edges are evenly distributed on the overlay and exist between high-bandwidth peers with a greater probability than between low-bandwidth peers. This applies to both intra- and inter-layer edges. Third, between peers in the same layer, the probability that an edge exists is inversely proportional to the distance. The lower the layer, the larger the impact of distance is.

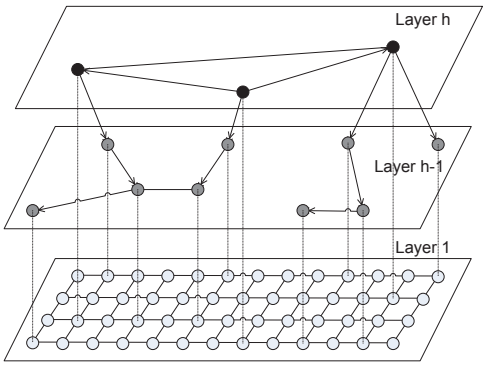


Fig. 1. Hierarchical overlay. Higher layers contains peers with higher bandwidth. Layer 1 contains short edges; upper layers contain rewired edges.

1: $tab \leftarrow \phi$
2: **for** j in rep **do**
3: **if** $prob(i,j) * RAND\_MAX > rand()$ **then**
4: $tab.insert(j)$
5: $rep.sort\_by\_cost(i)$
6: **for** j in rep **do**
7: **if** $tab.size() \geq TAB\_SIZE$ **then** break;
8: **if** $j \notin tab$ **then** $tab.insert(j)$

Fig. 2. Tracker's algorithm to compose a membership table for a peer i

On the hierarchical overlay, we desire a multicast tree similar to the one shown in Fig. 1. Chunks first go from the video server to a layer l peer, then spread along layer l edges to every region in the system. At lower layers, chunks spread using shorter edges. The last few hops, which determine the tree cost, are between nearby peers. Note that even for peers at high layers, most children are still nearby.

B. Overlay Construction

We assume the existence of a network positioning scheme. Peers report their positions and bandwidth to the tracker upon joining the system. The tracker maintains a repository of size N, which should be large enough to be a representative sample

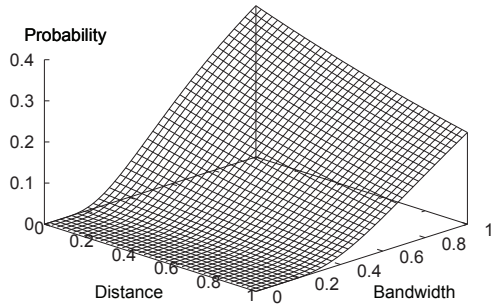


Fig. 3. Normalized probability that a long edge exist between two peers. The x and y axes are normalized bandwidth and distance between the two peers, $\beta = 1, \kappa = 2$

of the whole population, i.e., given a peer, the distribution of peers in the repository is the same as in the population with respect to the distance to the peer and upload bandwidth. When contacted by a peer i, the tracker uses the algorithm in Fig. 2 to compose a membership table for peer i and possibly adds peer i to its repository. In addition to nearby peers (lines 5–8), the tracker selects each remote peer j with probability p, where γ, β and κ are constant parameters (lines 2–4).

$$p = \frac{\gamma}{N} e^{-\frac{1}{\beta u} - \frac{d}{\kappa u}}$$

The proposed formula is designed to simultaneously achieve the properties described in III-A. An exponential function is adopted for two reasons. First, it has the desired attenuation property. We desire the probability to attenuate sharply when bandwidth or distance is large and flattens afterwards. Second, the pair-wise distances between peers on the Internet exhibit an exponential distribution [13]. Using an exponential function can retain the property on the overlay. Normalizing the upload bandwidth and distance as $u = \frac{(u_i - u_{min})(u_j - u_{min})}{(u_{max} - u_{min})^2}$ and $d = \frac{d_{ij}}{d_{max}}$ allows flexible definition of cost and bandwidth.

The factor $\frac{1}{N}$ is introduced to ensure that the number of rewired edges in the overlay is linear to the population. Parameter γ controls the fraction; a larger γ results in a larger fraction of rewired edges. To add more intra- and inter-layer edges between high-bandwidth peers, we introduce the factor $e^{-\frac{1}{\beta u}}$. The factor $e^{-\frac{d}{\kappa u}}$ ensures that (1) at any layer, the probability that an edge exists between two peers is inversely proportional to their distance, and (2) with lower bandwidth, distance has larger impact on the probability. We consider two factors to set the parameter β. A smaller β results in more uniform distribution of high-bandwidth peers in the overlay; a larger β leads to smaller tree height. When peers' upload bandwidth has uniform distribution, a value between 0.8–1.5 for β provides good trade-off. The parameter κ controls the rate that the probability grows with decreasing distance. The proposed formula is less sensitive to κ than to γ or β. A value between 1–10 for κ provides good trade-off. Fig. 3 shows the normalized probability distribution when $\beta = 1, \kappa = 2$.

The desired properties are maintained in the presence of peer churn as long as peers' arrivals and departures are independent from their bandwidths and positions in the substrate network. A value of several thousands for N is representative enough, the algorithm has complexity of $O(N(\lg N + 1))$, and the workload can be handled by a PC.

IV. Tree Building and Pull Mechanism

We briefly describe the tree construction, pull mechanism, and system architecture in this section. The system includes a server, a tracker, and a number of peers. Each peer maintains a membership table. A new peer, after obtaining its membership table from the tracker, randomly selects peers in the table to neighbor with. A peer with bandwidth u_i maintains $floor(1.5u_i)$ neighbors with a minimum of 4 (bandwidth is in units of the streaming rate in this paper). Each peer buffers 5 seconds of consecutive chunks before starting to play. The media player blocks at a missing chunk, or skips to the next available chunk if the peer has 2.5 seconds of consecutive chunks after the missing chunk. We use the heuristic DBSPT algorithm in [11] with one extension. On the hierarchical overlay, a DBSPT has the desired shape shown in Fig. 1. The new extension helps to balance upload bandwidth distribution in the system. If a peer cannot find parents, the peer disconnects some neighbors that have no spare bandwidth for certain time and finds new neighbors (i.e., the overlay is restructured). We use an innovative pull mechanism that is quick at detecting late chunks and has a high success rate of fetching them. Each peer maintains a sliding window of late chunks. A peer monitors chunk arrivals to estimate arrival times of future chunks. A chunk is considered late if it has not arrived 1 second after its due time. Every interval of length T_{pl}, a peer pulls all the late chunks in order of urgency (with respect to playback deadlines). Peers advertise their residual bandwidth to neighbors in buffer map messages. The scheduling algorithm attempts to maximize the number of chunks pulled while keeping load balanced among neighbors.

V. Performance Evaluation

A. Performance Metrics

Video quality is measured by the chunk delivery rate or mean time between glitches (MTBG). If a chunk fails to arrive before its playback deadline, the media player either freezes or skips, causing a *glitch*. A MTBG of 10 minutes is equivalent to a chunk delivery rate of 99.96%. Network cost is measured by the average tree edge cost. The playback delay refers to the time elapsed from when a chunk appears at the video server to when the chunk is played at a peer. When measuring multicast tree height, we only consider chunks that have reached more than 90% of the population. Because of peer churn, each chunk takes a set of different paths, which we consider as a *chunk tree*. We average metrics across all the valid chunk trees.

B. Simulation Setting

We use GT-ITM [14] to generate 10 networks with 40050 routers and about 200,000 links, then randomly connect 3000 peers to routers. Router-router link costs are assigned by GT-ITM and range from 1 to 3000; router-peer links have a fixed cost of 1. An overlay edge's cost is the cost of the shortest

substrate path; the propagation delay is linear to the cost and normalized to an average of 100 ms. The large numbers of routers and links reflect the complexity of the Internet; the delay distribution is similar to a measurement study [15]. We compare three overlays: random overlay, small-world overlay, and hierarchical overlay (abbreviated as RO, SO, and HO hereafter); the latter two have the same fraction of rewired edges (about 4%). We use two settings for peers' upload bandwidth—uniform distribution over range 1–4 ($U(1,4)$) and over range 1–9 ($U(1,9)$)—to investigate peers' behavior under loose and tight network constraints, respectively.

Peers maintain a buffer of 600 chunks. The chunk size is 16 KB; the streaming rate is 512 Kb/s. The intervals that peers exchange buffer maps and check whether to pull late chunks or switch parents are all 1 second. We start the interval timers of peers with a random offset to emulate the practical setting where peers are asynchronous. All peers join at time 0 (as a "flash crowd"); the video server starts streaming at time 5; the simulation ends at time 600. We set peer churn rate to 10% of the population per minute according to [7]. From time 10, one peer is turned off every 0.2 second until 600 peers are turned off at time 130. Then every 0.2 second, one peer is turned off and one previously turned off peer is turned on. Statistics are collected on the 1500 peers that have never been turned off. The overlay graph at time 5 is dumped for reference. Each test is run 10 times using the 10 substrate networks.

C. Simulation Results

1) Performance Under Loose Network Constraints: Fig. 4a shows that the heights of SPTs and chunk trees are smaller on HO than on SO, and the gap between chunk trees is greater than the gap between SPTs. The reason is that rewiring edges incident to large-degree peers results in a smaller diameter than rewiring randomly, and long edges play a more important role in spreading chunks across the overlay in HO (see Fig. 1) than in SO. The chunk trees' height remains at the same level throughout the simulation and is not impacted by peer churn.

Fig. 4b shows the average tree edge cost in HO and SO is about $\frac{1}{3}$ of that in RO. HO has slightly larger cost than SO because it uses more long edges to spread chunks. Fig. 4c shows that more chunks are lost in SO than HO and RO (discuss later). Because peers buffer 5 seconds before starting to play, the playback delay differs only slightly (see Fig. 5a); chunks can reach most peers in 1 second.

2) Impact of Long Edges: The fraction of long edges has great impact on the overlay. Rewiring 1% of edges to random nodes in a regular graph can reduce the characteristic path length by 80% [12], but a larger fraction is necessary to prevent partitioning in the presence of peer churn. Fig. 5b shows that the tree cost is proportional to the fraction but the tree heights decrease marginally when the fraction is larger than 15%. Rewiring 5–10% long edges usually provides good trade-off.

3) Performance Under Tight Network Constraints: Fig. 6a show that tree heights under tight network constraits have similar pattern as under loose network constraints but the values and gaps are larger because peers have smaller out-degrees. Fig. 6b shows HO and SO have a similar average tree edge cost, which is smaller than in Fig. 4b. When peers have smaller out-degrees and the overlay has fewer edges, long edges are less used because peers have less choices.

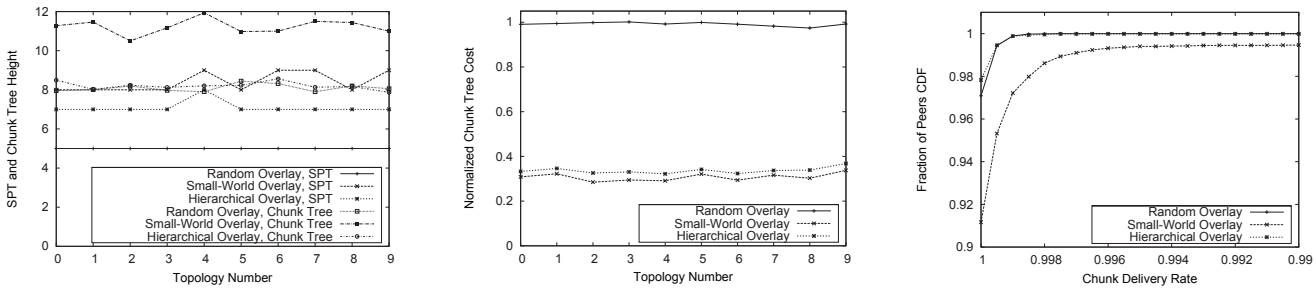
Fig. 6c shows that much fewer peers receive all the chunks when peers have less bandwidth, and the correlation between tree height and chunk losses is more obvious than in Fig. 4c. In our simulation, the substrate network and peers have no errors, and no chunks are lost when we simulate without peer churn, hence chunk losses are caused by peer churn. When a peer has a late chunk c due to peer churn, most descendants have to pull chunk c because the short buffering time before playback. A smaller upload bandwidth results in a lower pull success rate. Two factors contribute to the different chunk delivery rates in the three overlays. The primary factor is the different tree heights. Assume N homogeneous peers form a d-ary tree. Each peer has probability p to depart, and the departure causes e late chunks in each descendant. Given a pull failure rate f, on average, each peer will miss $O(pefh)$ chunks, where h is the tree height. The secondary factor is the distributions of chunk c and peers' spare bandwidth in the system. If both distributions are uniform, an arbitrary peer will have a higher probability to obtain chunk c from a neighbor. Both distributions are uniform in RO; because chunk c spreads via long edges in HO, it is more evenly distributed in HO than in SO.

When peers have less bandwidth, the gaps of playback delay between the three overlays becomes larger (see Fig. 5a) for two reasons. First, the gaps in terms of hops are larger. Second, the queueing delay at each peer is larger. In fact, queuing delays are larger than propagation delays, which only accounts for 13%, 17%, and 34% of the total delay of delivering packets from the server to peers in SO, HO, and RO respectively. SO has the longest delay because it has the largest tree height.

4) Trade-Off Between Delay and Delivery Rate: Buffering more chunks before playing improves the chunk delivery rate. According to [9], [11], 1–2% of chunks are pulled in hybrid schemes. Because all the late chunks will be pulled, the delivery rate is determined by the pull success rate. Fig. 5c shows that while a longer buffering time translates into a longer playback delay, the fraction of users receiving all the chunks becomes higher. We remark that further increasing the buffering time only marginally improves chunk delivery. Fig. 5c also shows that, when the overlay has fewer edges, more edges should be rewired. New chunks can spread faster and reach even distribution in the system in a shorter time.

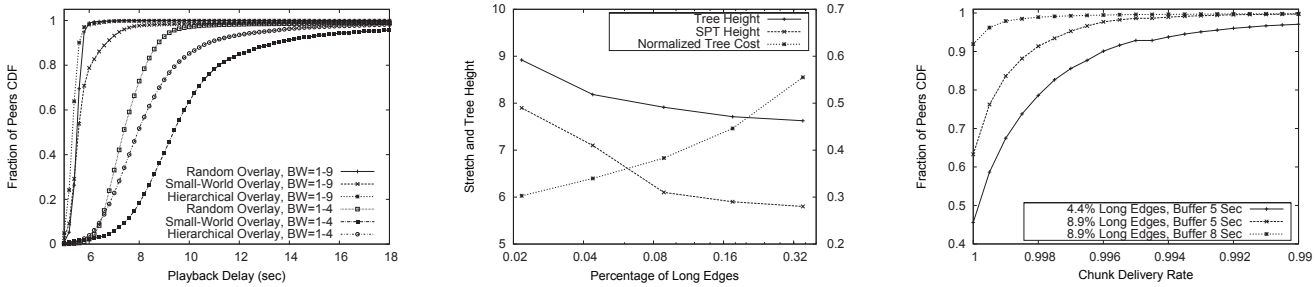
VI. Conclusion

Simply applying the neighbor-with-nearby-peers strategy to reduce network traffic in P2P video live streaming systems has a number of drawbacks, especially in data-driven hybrid schemes. In this paper, we propose a hierarchical overlay construction scheme, which has a small average edge cost and helps the DBSPT algorithm to build a short and robust tree. Simulation results under both loose and tight network constraints show that the hierarchical overlay outperforms the



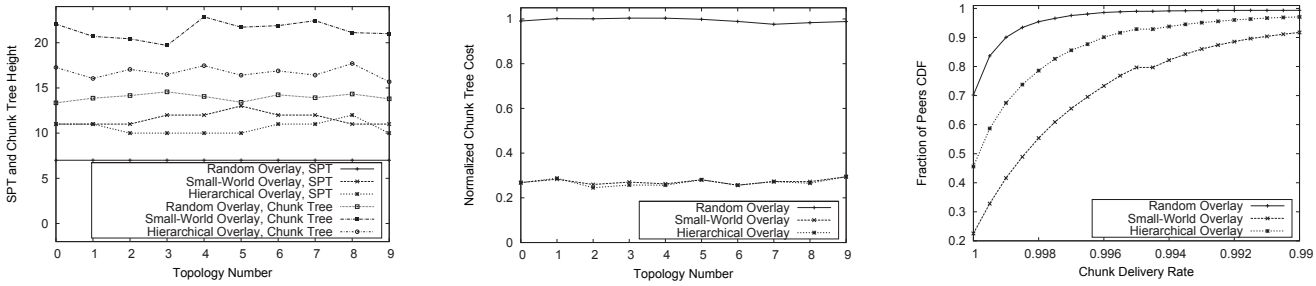
(a) SPT and average chunk tree height (b) Normalized chunk tree edge cost (c) Delivery-rate-user-ratio

Fig. 4. Performance when bandwidth setting is $U(1, 9)$.



(a) Playback delay CDF (b) Impact of fraction of rewired edges (c) Impact of playback delay

Fig. 5. Playback delay, impact of rewired edges and playback delay



(a) SPT and average chunk tree height (b) Normalized chunk tree edge cost (c) Delivery-rate-user-ratio

Fig. 6. Performance when bandwidth setting is $U(1, 4)$.

small-world overlay, and achieves performance similar to the random overlay but has only $\frac{1}{3}$ of the network cost.

REFERENCES

[1] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proc. IEEE*, vol. 96, no. 1, pp. 11–24, 2008.

[2] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?" *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, p. 40, 2007.

[3] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for P2P applications," in *Proc. ACM SIGCOMM Conf. on Data Communication*, 2008, pp. 351–362.

[4] D. Choffnes and F. Bustamante, "Taming the torrent," in *Proc. of ACM SIGCOMM*, 2008.

[5] T. Ng and H. Zhang, "Towards global network positioning," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001, pp. 25–29.

[6] X. Zhang and H. Hassanein, "On network utilization of peer-to-peer video live streaming on the internet," *to appear, Proc. ICC*, 2011.

[7] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "Coolstreaming: Design, theory and practice," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1661–1671, 2007.

[8] F. Wang, Y. Xiong, J. Liu, B. Burnaby, and C. Beijing, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *Proc. ICDCS*, 2007, p. 49.

[9] M. Zhang, L. Zhao, Y. Tang, J. G. Luo, and S. Q. Yang, "Large-scale live media streaming over p2p networks through global Internet," in *ACM Workshop on Advances in P2P Multimedia Streaming*, 2005, pp. 21–28.

[10] A. Ouali, B. Kerherve, and B. Jaumard, "Revisiting Peering Strategies in Push-Pull Based P2P Streaming Systems," in *IEEE International Symposium on Multimedia*, 2009, pp. 350–357.

[11] X. Zhang and H. Hassanein, "Treeclimber: A network-driven push-pull hybrid scheme for peer-to-peer video live streaming," in *Proc. IEEE Local Computer Networks*, 2010, pp. 372–375.

[12] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[13] E. Zegura, K. Calvert, S. Bhattacharjee *et al.*, "How to model an internetwork," in *IEEE infocom*, vol. 2, 1996, pp. 594–602.

[14] "http://www.cc.gatech.edu/projects/gtitm/," Accessed Mar. 2010.

[15] "http://www.cs.cornell.edu/people/egs/meridian/," Accessed Mar. 2010.