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

On Network Utilization of Peer-to-Peer Video Live Streaming On the Internet

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*—Efficient network utilization is essential towards the deployment of large scale peer-to-peer (P2P) video live streaming services on the Internet. Push-pull hybrid schemes and pull schemes provide practical solutions to P2P video live streaming, but network utilization remains unaddressed. In this paper, we use "typical" hybrid and pull schemes to investigate network utilization and whether the neighbor-with-nearby-peers strategy is applicable to P2P video live streaming applications. We find hybrid and pull schemes have limited capability in choosing low cost overlay edges although hybrid schemes are slightly better. The chunk tree height in hybrid schemes is significantly larger than in pull schemes. The neighbor-with-nearby-peers strategy reduces the chunk tree cost; however, it also results in a significantly larger tree height, lower reliability, and is less robust against peer churn, especially in hybrid schemes.**

I. Introduction

Peer-to-peer (P2P) video live streaming applications have gained great popularity in the past few years. In the same spirit of IP multicast, early P2P video live streaming schemes organize peers into multicast trees, usually with the aim of optimizing a cost function, to push video packets to each peer [1]. These *application layer multicast* (ALM) or *push* schemes have a short playback delay and use the Internet sparingly, but are vulnerable to peer churn and thus unsuitable in practical systems. Inspired by BitTorrent, recent schemes use the swarming technique. In these *pull* schemes [2], [3], peers advertise buffer map messages, which describe the video chunks they have, and pull missing chunks from one another. Pull schemes are robust against peer churn but have a long playback delay. Push-pull *hybrid* schemes [4]–[7] combine push and pull mechanisms together. In addition to pull missing chunks, peers also subscribe to neighbors to push future chunks. Since chunks are mostly pushed along the *subscription tree*, hybrid schemes have a short playback delay while being robust against peer churn. Unlike ALM schemes that build multicast trees to optimize network metrics, these pull and hybrid schemes [2]–[7] are *data-driven*, i.e., peers choose the neighbors to pull or subscribe according to the availability of chunks at neighbors. Since a peer's network position is unrelated to its chunks, enormous traffic is generated.

Network utilization is the primary objective in the design of many Internet protocols and applications, and is essential to the deployment of large-scale P2P video live streaming systems. Many ISPs limit P2P applications because of their enormous traffic and irresponsible use of the Internet. This is detrimental to video live streaming applications, since they are sensitive to packet loss. Understanding P2P video live streaming traffic is of utmost importance to control it, addressing both ISPs' need to reduce traffic and users' need to use the popular service. In this paper, we investigate network utilization in P2P video live streaming systems and attempt to answer following questions:

- A peer can potentially communicate with most peers in a P2P system, but will peers choose low cost edges to disseminate video packets?
- Given an overlay, will peers choose short paths (in terms of hops or cost) to the video server?
- The neighbor-with-nearby-peers strategy has been proposed for P2P file sharing systems to reduce traffic. Is it applicable in the context of P2P video live streaming?
- Will P2P video live streaming ever achieve network utilization close to IP multicast?

To the best of our knowledge, a study of network utilization in the context of P2P video live streaming does not exist in the literature.

The data-driven process in pull and hybrid schemes relies on the relative chunk delivery delays at neighbors to pull chunks or construct subscription trees. Since peers have no access to substrate link costs, and overlay edges' propagation delays and costs are strongly correlated, ALM and ISP-friendly P2P file sharing schemes usually use round trip times (RTTs) as an alternative. We use a similar approach. We have implemented "typical" hybrid and pull schemes (denoted TH and TP). TH constructs the subscription tree in a straight-forward manner—each peer subscribes to the neighbor with the most advanced proceeding position in the video. TP pulls urgent and rare chunks, and randomly selects neighbors that have the chunks and spare upload bandwidth to pull chunks from. We use two types of overlays: a *random overlay* where peers neighbor with random peers, and a *nearby overlay* where peers neighbor with nearby peers. We note that, in any scheme, the set of paths traversed by a chunk form a tree, which we term *chunk tree* and use it for evaluation. Some of the findings are as follows:

- Given an overlay, peers will not choose low cost edges in a data-driven scheme, although chunk trees have slightly lower costs in hybrid schemes than in pull schemes.
- Given an overlay, pull schemes construct a short tree if parameters are appropriately set. By contrast, hybrid schemes always have a significantly large tree height.

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

- The neighbor-with-nearby-peers strategy reduces the tree cost. However, it also results in higher trees, longer playback delay, lower reliability, and is less robust against peer churn. These side effects are more severe in hybrid schemes. In addition, the overlay is prone to partitioning.

The remainder of this paper is organized as follows. Section II introduces related work. Section III describes evaluation methodology. Section IV presents simulation results. Section V concludes the paper.

II. Related Work

P2P video live streaming schemes have two steps: peers first construct an overlay then pull chunks or build multicast trees on the overlay. P2P video live streaming schemes can be classified into *push*, *pull*, and *hybrid* schemes [1]. Both pull and hybrid schemes have been used in practical systems, and both split the video into fixed-length chunks. Pull schemes [2], [3] use similar chunk scheduling algorithms: peers pull urgent and rare chunks. They differ in pull target selection. In [2], a peer has a fixed number of neighbors and randomly selects a neighbor that has spare upload bandwidth. In [3], the number of neighbors a peer has is proportional to the peer's upload bandwidth; a peer randomly selects a neighbor to which the traffic rate is under a predefined limit. Hybrid schemes [4]–[7] have similar pull mechanism to recover missing chunks. They differ in tree building algorithms. Similar to pull mechanism, tree construction is data-driven, hence the tree is robust in the presence of peer churn. When a peer leaves, its children simply subscribe to other neighbors. In [4], peers consider neighbors' proceeding positions in the video to select parents. A similar approach is proposed in [5], but peers prefer "older" neighbors because they tend to stay longer in the system. In [6], peers consider the traffic volume they have received from neighbors in selecting subscription targets; all the peers must have synchronized clocks.

Most pull and hybrid schemes [2]–[5] construct a random overlay for its high connectivity and small diameter. In [6], a peer selects some neighbors with short RTTs from its membership table. However, since the membership table has a small size compared to the system population, the probability of selecting nearby peers is small. Reference [7] is similar to [6] (use the same simulator). New peers select parents using a proximity function with the aim of reducing playback delays. The scheme does not consider peer churn.

Several ISP-friendly schemes [8], [9] for P2P file sharing systems propose that peers only neighbor with nearby peers to construct the overlay with the aim of reducing cross autonomous systems (AS) traffic. These schemes differ in the technique to find nearby peers. This paper investigates whether this strategy is applicable to P2P video live streaming applications and use existing network positioning techniques.

III. Evaluation Methodology

A. Performance Metrics

We term a peer's path to the tree root along a chunk tree its *root path* and say a path is short or long depending on its hop count. Network utilization is evaluated using the average tree edge cost and *stretch*. The former is the tree cost divided by the number of tree nodes and makes fair comparison when a chunk reaches only part of the population. The latter, defined as the ratio of a peer's root path cost over the cost of its substrate IP unicast path to the video server, is widely used in ALM schemes to evaluate the similarity of overlay and substrate paths. Note that IP multicast protocols construct a shortest path tree (SPT). Although a minimum spanning tree (MST) has the lowest tree cost, it is prone to partitioning and forwarding loops, and has a large tree height.

TABLE I
System Parameters

Notation	Comment	Default
N	The number of peers in the system	3000
T_{bm}	Buffer map advertisement interval	1 sec
T_{sc}	Subscription check interval	1 sec
T_{pl}	Pull check interval	1 sec
W_{buf}	Buffer size (chunks)	600
W_{urg}	Emergency window size (hybrid/pull)	9/5
W_{swa}	Swarming window size (hybrid/pull)	0/240
W_{bbp}	Number of chunks to buffer before playing	20
W_{bbs}	Number of chunks to buffer before skipping	10

Video quality is evaluated by the chunk delivery rate, which is the fraction of chunks that arrive at a peer before playback deadlines, or mean time between glitches (MTBG). Assume each lost chunk causes a glitch, a MTBG of 10 minutes is equivalent to a chunk delivery rate of 99.96%. Compared with audio streaming, video streaming is sensitive to chunk losses [1]. Most users require smooth playback or do not use the service at all. Departures of unsatisfied users cause more lost chunks at satisfied users, resulting in a chain reaction. 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. It is the sum of the chunk delivery delay and chunk buffering delay (i.e., the time chunks are buffered at the peer played).

Chunk losses are either substrate- or overlay-related. We use the term *reliability* to describe chunk losses caused by errors of substrate network elements. A long root path is less reliable because more substrate elements are involved, hence the chunk tree height is an indicator of reliability. Overlay-related chunk losses are caused by pull and push mechanism, especially in the presence of peer churn. In this paper, we assume the substrate network is reliable, all the errors caused by links transmission, routers dropping packets, and multi-tasking hosts temporarily blocking the P2P client are ignored.

B. Typical Pull (TP) and Hybrid (TH) Schemes

TP and TH have the same bootstrap, overlay construction, and buffer management mechanism, which are similar to other pull and hybrid schemes. The system contains a server, a

[1]Multiple description coding (MDC) is often proposed to conceal chunk losses in P2P video streaming schemes. It is rarely used in practical systems due to its inefficiency.

tracker, and a number of peers. A new peer obtains the IP addresses of the video server and tracker by browsing a web page. Each peer maintains a membership table. A new peer obtains its initial membership table from the tracker and may contacts the tracker for more peers when necessary. The particular positioning scheme for locating nearby peers is irrelevant to our evaluation. We simply assume the tracker has global knowledge of the system. The tracker selects random peers to compose a membership table for the requesting peer when constructing a random overlay, and selects nearby peers when constructing a nearby overlay. After obtaining its membership table, a peer randomly selects peers from the table to neighbor with. The number of neighbors a peer maintains is proportional to its upload bandwidth. The video is split into chunks of fixed size. Each chunk has a unique sequence number. Each peer maintains a buffer of size W_{buf} to hold recently received chunks. Every interval of length T_{bm}, each peer advertises its buffer map to neighbors. The message consists of the latest chunk's sequence number n and a vector, whose ith element indicates whether the peer has chunk $n-i$. The buffer management module feeds chunks to the media player. We simulate a commercial media player as follows. If the media player is about to play chunk c but the chunk is unavailable, it will block. It will resume playing chunk c when the chunk arrives or skip to the next available chunk if the peer has W_{bbs} consecutive chunks after chunk c. System parameters are shown in Table I.

As in other pull schemes, the buffer is sequentially divided into two windows in TP: the beginning part is called emergency window, which consists chunks approaching playback deadlines; the rest is called swarming window. Every interval of length T_{pl}, peers pull all the chunks in the emergency window and the rarest chunks in the swarming window. Peers only pull from neighbors with spare bandwidth. This approach, also used in [2], has a higher pull success rate than [3]. Assume a peer wants to pull m chunks from n neighbors, and each neighbor has certain chunks. The matching process starts with chunks that are available at only 1 peer, then chunks available at 2 peers, and so on. In each step, urgent chunks are matched before rare chunks, and a random target is selected if multiple qualified neighbors exist. A new peer first swarms until $\frac{3}{4}$ of its swarming window is filled; then it starts playing after receiving W_{bbp} consecutive chunks.

In TH, most chunks are pushed. A peer pushes a chunk to its children immediately after receiving it. Peers only pull urgent chunks and use the same target selection algorithm as in TP. The subscription tree is constructed in a straightforward data-driven manner. Every interval of length T_{sc}, peers attempt to switch parents. Neighbors are ranked according to their proceeding positions in the video. The neighbor with the most advanced position is the candidate. A peer' child cannot be its candidate parent. To avoid frequent switching, a peer switches parents only when the candidate has a margin of 1 second. A child must subscribe to its parent periodically to keep its status as a child. Peers employ connection admission control (CAC) when handling subscription requests. A peer accepts children only if it has spare bandwidth. A peer guarantees the bandwidth to its children; the pull mechanism uses the residual bandwidth. A new peer starts playing after having W_{bbp} consecutive chunks.

C. Simulation Setting

We first use the transit-stub model of GT-ITM [10] to generate 10 substrate networks with 40050 routers and about 200,000 links, then randomly connect 3000 peers to routers. The cost of router–router links is assigned by GT-ITM and ranges from 1 to 3000; the cost of peer–router links is 1. An overlay edge's cost is the same as the cost of the shortest IP path between the two peers. Overlay edges' propagation delays are set to be linear to their costs and normalized to 100 ms. To investigate peers' behaviour under loose and tight network constraints, we set peers' upload bandwidths with two settings: uniform distribution over range 1–9 ($U(1,9)$) and over range 1–4 ($U(1,4)$). (Bandwidth are in units of the streaming rate in this paper.) With the larger setting, the overlay has more edges, and peers have more choices in selecting pull or subscription targets; with the smaller setting peers behavior is more constrained by the network. A peer with bandwidth u_i maintains $floor(1.5u_i)$ neighbors with a minimum of 4.

The streaming rate is 512 Kb/s. The chunk size is 16 KB. Peers exchange buffer maps, check whether to pull late chunks or switch parents every 1 second. To emulate a practical setting where peers are asynchronous, we start peers' interval timers with a random offset. In TP, peers have a swarming window of 240 chunks and an emergency window of 5 chunks. In TH, peers do not swarm but needs a larger emergency window to recover missing chunks. Parameter are shown in Table I.

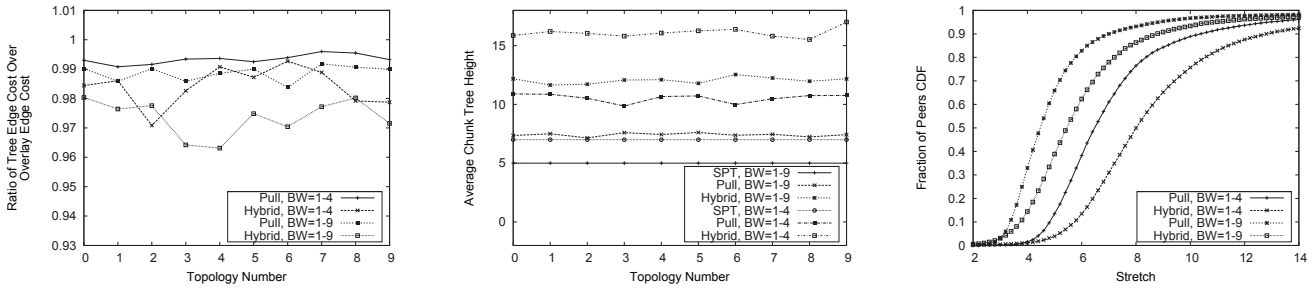
All peers join the system at time 0 to emulate a "flash crowd", the video server starts streaming at time 5, and the simulation ends at time 600. According to the measurement results in [4], we set peer churn rate to 10% of the population per minute. From time 10, one peer is turned off every 0.2 second 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 and from the 80-th chunk to the 400-th chunk from last (because pull schemes have a long playback delay). At time 5, we dump the overlay graph obtain its SPT's height. Each test is run 10 times using the 10 substrate networks (and confidence intervals are calculated).

IV. Evaluation Results

In this section, we first present results on the random overlay, which reflect network utilizations of existing P2P schemes. Then using the nearby overlay, we investigate the impact of the neighbor-with-nearby-peers strategy.

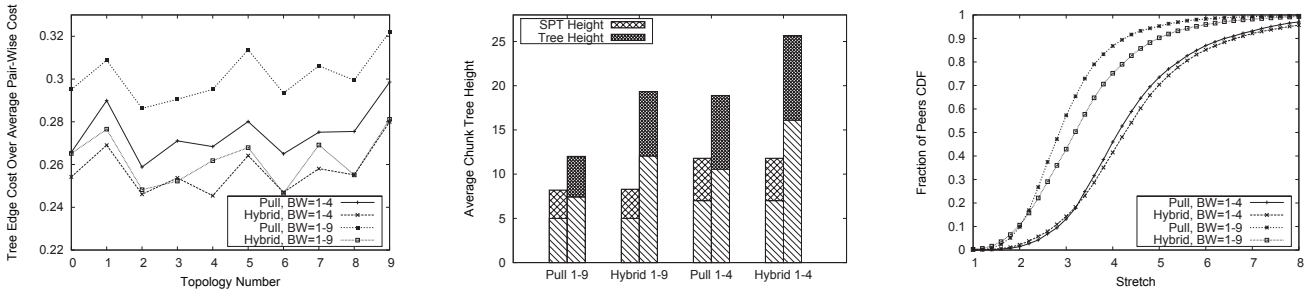
A. Results Using Random Overlay

The average edge cost of the random overlay is statistically equal to the average of all the pair-wise costs between peers. Fig. 1a shows that chunk trees in both TH and TP have a cost



(a) Ratio of tree edge cost over overlay edge cost (b) Chunk tree and SPT height (c) Peers' stretch

Fig. 1. Chunk tree's cost and height and peer's stretch on the random overlay



(a) Ratio of tree edge cost over pair-wise edge cost (b) Chunk tree and SPT height (c) Peers' stretch

Fig. 2. Chunk tree's cost and height and peer's stretch on the nearby overlay. The slanted boxes in (b) correspond to the random overlay.

close to a random tree on the overlay. This result suggests that the costs (or propagation delays) of overlay edges have little impact on peers' decision to select subscription or pull targets. TH has a slightly lower average tree edge cost than TP; the gap is small but consistent.

Fig. 1b shows the average chunk tree height (averaging across all the chunk trees that reach 90% or more of the population). Since statistics are collected on the 1500 peers that have never been turn off, the reported chunk tree height is a close approximation of the actual tree height. TH has higher chunk trees than TP, and the gap is larger when peers have lower bandwidth. The tree height is 12.0 and 16.1 for TH and 7.4 and 10.6 for TP when the bandwidth setting is $U(1,9)$ and $U(1,4)$, respectively (the 95% confidence interval is within ± 0.3). Considering the system has only 3000 peers, the difference is significant.

A peer's out-degree on chunk trees is constrained by its upload bandwidth. The optimal tree height is unavailable since the degree bounded shortest path tree (DBSPT) problem is NP-hard. We plot SPT's height in Fig. 1b for reference. We find that many nodes on SPT have large out-degrees, so we estimate TP's chunk tree height is close to optimal [2]. In TP, the delivery delay of a chunk is strongly correlated with its hop counts from the source, so peers have a high probability to pull chunks from a neighbor that has fewer hops to the source. In TH, the delivery delay is loosely correlated with hop counts, and is usually smaller than the subscription check interval T_{sc}, thus chunk trees are tall.

Since chunk tree and overlay edges have similar cost, TH and TP have large stretch (see Fig. 1c). The 95% confidence interval is 6.1–6.9 and 5.0–5.7 for TH and TP respectively when the bandwidth setting is $U(1,9)$. TH has higher stretch than TP because TH has a larger chunk tree height.

B. Results Using Nearby Overlay

We first let peers randomly select from the closest 1% of peers to neighbor with; all the overlays are partitioned. To reduce partitioning while keeping low overlay edge costs, we rewire a small number of edges to random peers to form a "small-world" graph. According to [12], rewiring 1% of edges in a regular graph reduces the characteristic path length by 80%. To keep the overlay connected in the presence of peer churn, we rewire 5%.

Fig. 2a shows that the average chunk tree edge cost on the nearby overlay is $\frac{1}{4}$ to $\frac{1}{3}$ of that on the random overlay. The cost reduction is the result of the overlay construction rather than peers' subscription or pull target selection. In fact, chunk tree edges have an average cost 15–30% greater than overlay

[2]TP's chunk tree height is similar to that of a heuristic DBSPT algorithm presented in [11].

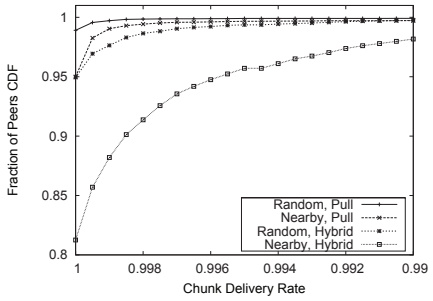


Fig. 3. Delivery-rate-user-ratio when the bandwidth setting is $U(1,9)$. The y-axis is the cumulative fraction of peers whose chunk delivery rate exceeds the value plotted on the x-axis.

edges. Rewired long edges are over-represented on chunk trees because they connect the otherwise partitioned cliques. As on the random overlay, TH exhibits better capability than TP in choosing low cost edges. When the bandwidth setting is $U(1,9)$, TP's tree edge cost is 15% greater than TH's. In TP, the propagation delay is negligible compared with the chunk delivery delay. In TH, the propagation delay account for a considerable part of the delivery delay and hence has more impact on peers' subscription decisions.

Fig. 2b contrasts tree height between the two overlays for TH and TP. First, the gap of SPT's height between the two overlays is significant. Second, the gap of the chunk tree's height between the two overlays is greater than SPT's height. Third, the gap of the chunk tree's height between TH and TP is greater on the nearby overlay or when peers have lower bandwidth. Although small-world and random graphs asymptotically have the same diameter of $O(\log N)$, the nearby overlay actually has a significant larger diameter. Because nearby overlay edges have smaller propagation delays, the propagation delay of a path is less correlated with its hop count than on the random overlay. Therefore, peers' subscription is more random.

Fig. 2c shows that the stretch on the nearby overlay is 50–60% of that on the random overlay. The 95% confidence interval is 3.2–3.9 and 3.0–3.5 for TH and TP when the bandwidth setting is $U(1,9)$. Recall that the average edge cost on the nearby overlay is $\frac{1}{4}$ to $\frac{1}{3}$ of that on the random overlay, increased tree height hinders the improvement of stretch.

The neighbor-with-nearby-peers strategy has significant negative impact on video quality. The fraction of peers that receive all the chunks (MTBG is greater than 8 minutes) in TH drops from 94.9% on the random overlay to 81.3% on the nearby overlay when the bandwidth setting is $U(1,9)$ as depicted in Fig. 3, and from 54.3% to 26.8% when the bandwidth setting is $U(1,4)$. Note that the substrate network has no errors, and all the peers receive all the chunks in TH when we simulate without peer churn. We conclude that chunk losses in TH are caused by peer churn. By contrast, TP performs only slightly better on the random overlay (3.9% more peers receive all the chunks). On the nearby overlay, chunks tend to propagate along a group of "similar" paths. In TH, because the playback delay is small, if a chunk is missing at a peer, it is likely the chunk is also missing at the peer's neighbors before the chunk's playback deadline at the peer. In TP, because the chunk delivery delay has a large variance, a large chunk buffering delay has to be used, which is large enough for the chunk to arrive at the peer's neighbors. Therefore, TP is robust against peer churn if and only if a large playback delay is allowed. On the random overlay, chunks may arrive at a peer via "diverse" paths. If a chunk is missing at a peer, it is likely that the chunk will have arrived at the peer's neighbors via other paths before the chunk's playback deadline at the peer. This path diversity of random overlay improves the robustness against peer churn in TH and TP.

V. Conclusion

Network utilization is essential to P2P video streaming applications but remains unaddressed in existing P2P video live streaming schemes. In this paper, we use typical hybrid and pull schemes to investigate the network utilization and whether the neighbor-with-nearby-peers strategy is applicable. We find pull schemes to have a tree cost similar to that of a random tree but can construct a short tree with proper configured parameters. Hybrid schemes have a smaller tree cost than pull schemes, but the chunk trees can be up to 60% higher. Using the neighbor-with-nearby-peers strategy reduces the tree cost, but also results a higher tree, longer playback delay, lower reliability against network element errors, and less robustness against peer churn, especially in hybrid schemes. The system is also more likely to be partitioned.

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] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE INFOCOM*, vol. 3, 2005, pp. 13–17.

[3] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *Proc. IEEE INFOCOM*, 2007, pp. 1415–1423.

[4] 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.

[5] 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.

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

[7] A. Ouali, B. Kerherve, and B. Jaumard, "Revisiting Peering Strategies in Push-Pull Based P2P Streaming Systems," in *11th IEEE International Symposium on Multimedia*, 2009, pp. 350–357.

[8] 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.

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

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

[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.