

Optimizing P2P Streaming Throughput under Peer Churning

Yi Cui, Liang Dai, and Yuan Xue

Department of Electrical Engineering and Computer Science, Vanderbilt University

Nashville, TN, 37235

{yi.cui,liang.dai,yuan.xue}@vanderbilt.edu

Abstract—High-throughput P2P streaming relies on peer selection, the strategy a peer uses to select other peer(s) as its parent(s) of streaming. Although this problem has been thoroughly investigated in the classical optimization framework, it still remains unaddressed as how to sustain throughput competitive to the optimum under highly dynamic peer churning. We propose an online distributed peer selection algorithm based on “shortest-path” routing where the overlay edge length is an asymptotic function of its traffic load. This basic algorithm is further extended under practical settings such as multi-parent streaming, admission control, delay constraint, and simplified topology. In all above settings, we prove approximation bound of our algorithm to the optimal throughput. Through evaluation under different topological setups and peer churning sequences, we show our solution to consistently deliver competitive throughput, which greatly outperform its theoretical bound.

I. INTRODUCTION

P2P streaming has been approved a highly cost-effective content distribution solution, where peers self-organize themselves into an overlay network and relay data to each other, thus reducing server load. A central problem in the overlay network construction is *peer selection*, the strategy a peer employs to select other peer(s) as its parent(s) of streaming. Under the static setting, it boils down to the overlay multicast problem: given the source (streaming server) and a fixed set of receivers (peers), how to set up the optimal spanning tree to interconnect them? Under different application contexts, there are various optimization objectives, such as maximizing throughput to all receivers, minimizing the average or worst-case delay, minimizing service disruptions, etc.

Although existing work has proved this problem hard to solve under the centralized and static setting, the realistic constraints of P2P streaming makes it even more challenging. The central challenge is *peer churning*, i.e., a peer can dynamically join or leave the streaming by the will of its user. This characteristics leaves a constantly changing peer set throughout the lifetime of P2P streaming, which makes any attempt to globally coordinate the peer selection prohibitively expensive, if at all possible. Therefore, existing P2P streaming systems choose to perform peer selection in an *online* fashion, where a newly joined peer autonomously selects one or multiple existing peers as its parent(s). If a peer is disconnected

due to its parent leaving, it will rejoin the streaming as a new peer.

However, an apparent gap exists between the online methodology and the earlier-stated optimization objectives, which require the optimal distribution structure to be setup from scratch, and torn down to restart again if any peer churning event happens. Intuitions learned from earlier works[1][2] also suggest global coordination of all peers.

Based on this observation, any attempt to bring optimization flavor into the practical online strategy should answer the following questions. If the optimization objective could not be achieved exactly, will the online solution perform to any competitive ratio to optimality? If so, does there exist a lower-bound that is guaranteed theoretically, and consistently sustained beyond experimentally? As the peer churning continues during the course of P2P streaming, will the online solution result in steady or deteriorating performance? Is reorganization strategy needed to reshape the distribution structure along the time, for the purpose of performance tuneup?

This paper sets out to answer these questions. In particular, we focus on the objective of *throughput maximization*, since it holds great practical value for P2P streaming practitioners to maintain or improve the quality of multimedia streaming. We prove that, under peer churning, our online peer selection solution is able to achieve logarithm competitive ratio to the optimal throughput. This claim still holds when we extend our solution and study its performance under various evaluation settings listed below.

- 1) *Multi-parent Streaming*: The basic solution only allows each peer to have one parent. We show that it can be upgraded to the case of multi-parent streaming, an approach adopted by many existing solutions[3], [4], [5].
- 2) *Admission Control*: In many P2P streaming applications, streaming rate is predetermined by the server, which calls for the admission control mechanism to reject peers unable to achieve the desired throughput. We show that our online solution could be easily adapted to this scenario.

The rest of this paper is organized as follows. Then in Sec. II, we introduce the network model and optimization framework. Sec. III presents the basic online peer selection algorithm, and its implementation issues. We discuss performance evaluation in Sec. IV and conclude in Sec. V. Due to space constraint, we delay the proofs to the theorems appeared in this paper to our technical report[6].

This work was supported by NSF Career Grant grant under contract number NSF CNS 0643488, Vanderbilt University Discovery Grant, and Microsoft Gift. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

II. PROBLEM FORMULATION

A. Overlay Graph Model

We view the network as a directed graph $G = (\mathcal{V}, \mathcal{L})$, with capacity c_l on each physical edge $l \in \mathcal{L}$. We are given a commodity set \mathcal{M} . Each commodity $m \in \mathcal{M}$ is an overlay session¹. $S(m)$ is the source, i.e., the server distributing the multimedia stream, and $\mathcal{R}(m)$ is the set of receivers. The application considered in this paper is assumed to employ the non-scalable media codec. Therefore, we define $dem(m)$ as the demand of m , which is the desired streaming rate from $S(m)$ to all receivers in $\mathcal{R}(m)$.

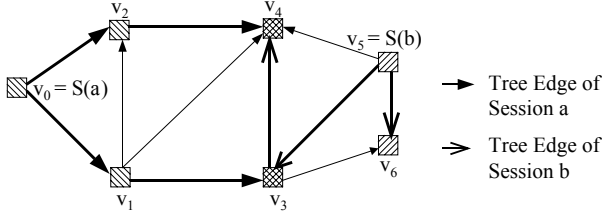


Fig. 1. Illustration of Overlay Spanning Tree

For a session $m \in \mathcal{M}$, we define its overlay graph $G(m) = (\mathcal{V}(m), \mathcal{E}(m))$. In this graph, the node set $\mathcal{V}(m) = \{S(m)\} \cup \mathcal{R}(m)$ includes the sender and all receivers of session m . A directed edge $(v_s, v_t) \in \mathcal{E}(m)$ represents the data dependency between two nodes v_s and v_t , i.e., v_t can retrieve data from v_s . In Fig. 1, two sessions, a and b , coexist in the same physical network. $\mathcal{V}(a) = \{v_0, v_1, v_2, v_3, v_4\}$. Here, peer v_0 is the sender of session a , and peers v_1 through v_4 are the receivers of a . For session b , $\mathcal{V}(b) = \{v_3, v_4, v_5, v_6\}$. Here, peer v_5 is the sender of session b , and the rest belong to the receiver set $\mathcal{R}(b)$. Each overlay edge, e.g., (v_0, v_1) in Fig. 1, corresponds to the unicast route at the physical network.

B. Solution Methodology

On top of each overlay graph, different spanning trees can be found, as highlighted in Fig. 1. With the formulation of overlay graph, the overlay routing problem boils down to, for each session $m \in \mathcal{M}$, finding a set of spanning trees on top of its overlay graph $G(m)$. The flow rate of session m is the aggregated rate of all its spanning trees. We collect these trees into the set $\mathcal{T}(m) = \{t_j(m)\}$. We use $f_j(m)$ to denote the flow of commodity m sent along the tree $t_j(m)$. Our goal is to maximize the flow rates of all sessions, formulated as the following linear programming problem.

M :

$$\text{maximize } f \quad (1)$$

$$\text{subject to } \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \geq f \cdot dem(m), m \in \mathcal{M} \quad (2)$$

$$\sum_{m \in \mathcal{M}} \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \cdot n_l(t_j(m)) \leq c_l, l \in \mathcal{L} \quad (3)$$

$$f \geq 0, f_j(m) \geq 0, 1 \leq j \leq |\mathcal{T}(m)|, \forall m \in \mathcal{M}$$

¹We will use the terms session and commodity interchangeably thereafter in this paper.

The objective of **M** is to maximize the scalar value f , subject to the fairness and capacity constraints. Inequality (2) enforces *fairness constraint* by requiring that the comparative ratio of traffic routed for different commodities satisfies the comparative ratio of their demands, i.e., at least $f \cdot dem(m)$ units of commodity flow can be routed simultaneously for each session $m \in \mathcal{M}$. Thus, the absolute value of $dem(m)$ is meaningless, as we can easily tune the value of f by scaling up/down all demands, while $f \cdot dem(m)$ stays unchanged. Inequality (3) specifies the *capacity constraint* that the traffic routed on each physical edge $l \in \mathcal{L}$ does not exceed its capacity c_l . Note here that $n_l(t_j(m))$ is an integer value denoting the number of appearances of l in $t_j(m)$. This value could be greater than one, since a physical edge may appear in a tree more than once, a unique feature of overlay network.

This problem, also known as “packing overlay spanning trees”[7], is proved by our previous work to be solvable in polynomial time. This means we can derive the optimal value of the scalar f , denoted as f^* .

The solution to obtain f^* is to assign an artificial length d_l to each physical link l , which is a function of the overlay traffic it carries. Starting from a small initial value, d_l asymptotically increases as l is assigned more traffic. On the other hand, traffic is routed through the minimum overlay spanning tree based on the above link length, which is by definition the most lightly-loaded route. The algorithm proceeds in alterations of “link cost update” and “traffic routing”. Interesting readers are referred to [7] for detailed presentation and analysis of the algorithm.

C. Discussion

Such an optimal algorithm relies on the following unrealistic assumptions. If they are compromised, so will be its optimality and the tractability of the problem.

First, any *peer churning* event will cause the established spanning trees to be torn down and rebuilt. Second, each peer is allowed to have *unlimited number of parents* in order to achieve the optimal rate. The problem becomes NP-hard[7] when one tries to limit the number of parents. Third, the optimal algorithm assigns length to each physical link, which requires *complete physical network knowledge*, i.e., each overlay session has the complete knowledge of its underlying topology, as well as the capacity of each physical link.

Nevertheless, this algorithm constitutes the methodological foundation to the online peer selection algorithm we are about to present. Furthermore, due to its ability to derive optimal throughput, our performance evaluation uses it as the baseline algorithm to evaluate the approximation of the online algorithm to the optimality.

III. ONLINE PEER SELECTION ALGORITHM

A. Overview

In this section, we present our online peer selection algorithm. The algorithm essentially follows the solution methodology elaborated in Sec. II-B. Here, the tree construction is decomposed into autonomous actions taken by each peer in response to different peer churning events. Our algorithm

consists of three elementary functions. **Join** is run by each new peer joining a P2P streaming session. **Leave** is the operation for each old peer leaving its session.

B. Basic Algorithm

Initialization	
1	$\forall l \in \mathcal{L}, d_l \leftarrow \beta/c_l, \sigma_l \leftarrow 0$
/* When Peer v_{new} Joins Session m */	
Join (v_{new}, m, dem)	
1	$p(v_{new}, m) \leftarrow \text{argmin}\{d(v, v_{new}) \mid (v, v_{new}) \in G(m)\}$
2	$\mathcal{R}(m) \leftarrow \mathcal{R}(m) \cup v_{new}$
3	$\forall l \in (p(v_{new}), v_{new})$
4	$d_l \leftarrow d_l(1 + \rho \frac{dem}{c_l}), \sigma_l \leftarrow \sigma_l + \frac{dem}{c_l}$
5	$d(p(v_{new}), v_{new}) \leftarrow \sum_{l \in (p(v_{new}), v_{new})} d_l$
/* When Peer v_{old} Leaves Session m */	
Leave (v_{old}, m, dem)	
1	$\mathcal{R}(m) \leftarrow \mathcal{R}(m) - v_{old}$
2	$\forall l \in (p(v_{old}), v_{old})$
3	$d_l \leftarrow d_l/(1 + \rho \frac{dem}{c_l}), \sigma_l \leftarrow \sigma_l - \frac{dem}{c_l}$
4	$d(p(v_{old}), v_{old}) \leftarrow \sum_{l \in (p(v_{old}), v_{old})} d_l$
5	$\forall v_{child} \in \{v_{child} \mid p(v_{child}) = v_{old}\}$
6	$\forall l \in (v_{old}, v_{child})$
7	$d_l \leftarrow d_l/(1 + \rho \frac{dem}{c_l}), \sigma_l \leftarrow \sigma_l - \frac{dem}{c_l}$
8	$d(v_{old}, v_{child}) \leftarrow \sum_{l \in ((v_{old}, v_{child}))} d_l$
9	Join (v_{child}, m, dem)

TABLE I
ONLINE PEER SELECTION ALGORITHM

We list the elementary functions of online peer selection algorithm in Tab. I. We introduce load indicator σ_l for each link $l \in \mathcal{L}$, which indicates the percentage of l 's capacity occupied by traffic.

During the initialization phase, an initial length β is assigned to each link, normalized by its capacity. Here, we note that there is no need to initialize all links in one shot. If no peer exists in the P2P session m , the physical link set \mathcal{L} remains empty. \mathcal{L} gradually grows as more peers join m . Therefore, a link l should stay uninitialized until the newly joined peer v_{new} introduces it into \mathcal{L} . Its length d_l and load indicator σ_l should be updated and maintained by v_{new} as well. More discussions will follow in Sec. III-C.

When a new peer v_{new} joins the session m , it runs the **Join** function to find its parent $p(v_{new})$, and attach itself to the found parent. According to the definition of the overlay graph $G(m)$ in Sec. III-A, an overlay edge in $G(m)$ is directed from a peer v to v_{new} if v can serve as the parent of v_{new} to retrieve data from. Therefore, all peers which have an overlay edge directed towards v_{new} are its candidate parents. Among these peers, our algorithm chooses the one with the minimum overlay edge length as the parent of v_{new} (Lines 1 and 2). Then for each physical link along the overlay edge $(p(v_{new}), v_{new})$, we route dem amount of traffic, record the change to its load indicator, and update its edge length (Lines 3 to 5) based on the following function.

$$d_l \leftarrow d_l(1 + \rho \frac{dem}{c_l})$$

Evidently, d_l increases in a super-linear fashion as the traffic load indicator σ_l increases. The length of an overlay edge

e is the aggregate length of physical links along its unicast route. Such a cost function definition aggressively increases the edge length of an occupied overlay edge, in order to direct peer selection to under-utilized overlay edges. The step size ρ controls the growth speed of the edge length. This attempt to achieve load-balancing helps maximizing capacity utilization, hence maximizing throughput.

If an old peer v_{old} leaves the P2P session m , then **Leave** function is called. In this function, the leaving peer v_{old} first terminates the incoming traffic from its parent, then stops feeding its children, if any (Line 1). v_{old} then reduces the lengths and load indicators of overlay edges connecting itself to its parent and children (Lines 2 to 8). Here, the algorithm reverses what was done in the **Join** function. Finally, the orphaned children of v_{old} is notified to find themselves new parents by rejoining the session m as a new peer.

C. Practical Issues

We now discuss several practical issues involved in the basic algorithm.

1) *Bootstrapping*: Upon joining a P2P streaming session, the new peer v_{new} should seek from a bootstrapping node the initial knowledge of the session. Most existing solutions[8], [9] choose the media server ($S(m)$ in the context of this paper) as the bootstrapping node (tracker server)², which essentially includes the set of candidate parents for v_{new} . To fulfill the server with such knowledge, v_{new} should report to $S(m)$ its presence after successfully joining the session.

2) *Traffic Scaling*: Both **Join** and **Leave** functions require the involved peers to specify dem , the *desired streaming rate*. Since we assume single-rate P2P streaming, we should set the desired rate as $dem(m)$ as defined in Sec. II. This is different to the *achievable rate*, which is the maximum throughput among all peers allowed by the physical network capacity. Formally, we denote it as $f_{achievable}$. If the load indicator $\sigma_l < 1$ for all physical links, then $f_{achievable} > dem(m)$, which means the streaming rate could be amplified to fully utilize the remaining capacity. If $f_{achievable} < dem(m)$, then at least one link is congested, i.e., $\sigma_l > 1$, which needs the streaming rate decreased. Traffic scaling is a useful mechanism if the streaming rate could be dynamically adapted at the data source, such as live event broadcasting.

3) *Loop Avoidance*: Each peer selects its parent from the set of candidate parents defined by the overlay graph $G(m)$. If $G(m)$ is a directed acyclic graph (DAG), there is a strong partial ordering of relaying dependency among peers. Here, a peer can only retrieve data from those which arrived earlier and already possessed the data, e.g., on-demand streaming³. However, the DAG restriction does not apply in certain applications such as teleconferencing and live broadcasting. In this case, a rejoining peer v_1 might choose one of its own descendants as the parent. As illustrated in Fig. 2 (a), if v_1 leaves, its child v_2 can choose v_3 or v_5 as its new parent. However, choosing v_5

²Due to the dual identity of $S(m)$, it will be referred to as either the media server or the bootstrap server, depending on the usage context.

³We note the DAG restriction can be easily sustained by the bootstrapping server by tagging each current peer with its joining time

will result in a loop which cuts v_2 away from the source $S(m)$. To enforce loop avoidance, either v_2 itself or the server $S(m)$ should keep the up-to-date view of all descendants of v_2 . In fact, a simple modification to the bootstrapping procedure can address this issue. After the new peer v_{new} joins the session, we ask it to report to $S(m)$ not only its presence, but also the parent it chooses. This enables $S(m)$ to have the global view of the entire streaming session. $S(m)$ can easily achieve loop avoidance by excluding rejoining peer descendants from its candidate parent set.

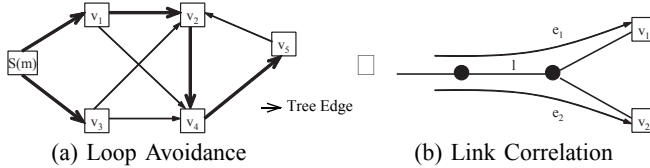


Fig. 2. Practical Issues in Online Peer Selection

4) *Link Capacity Measurement and Overlay Edge Length Computing*: In order to compute the length d_e of an overlay edge e , we need to know the capacity of each physical link along its unicast route, which can be measured by tools, such as pathchar[10], in an end-to-end manner. A directed overlay edge is solely owned by its two end peers, therefore should be managed by either its sender or receiver. We choose the receiver-based approach: since v_{new} invites e into the P2P streaming session, it should also update the initialization, updating, and maintenance of d_e .

Ideally, each peer only needs to keep to itself the above management job and incur no communication overhead. However, a unique phenomenon in P2P and overlay networks termed “link correlation” greatly complicates the problem. As an example shown in Fig. 2 (b), the link l is owned by both overlay edges e_1 and e_2 . To ensure accurate parent selection, a peer should stay updated about the current lengths of all physical links its overlay edge encompasses. Therefore, when v_1 routes new traffic through l , it should update this change to v_2 as well.

In addition, upon bootstrapping of v_2 , the server $S(m)$ is responsible to remind v_2 that another peer v_1 is sharing the link l with one of its overlay edges. This requires $S(m)$ to have the global knowledge of the underlying physical network encompassed by all peers it serves. In order to achieve so, the bootstrapping procedure needs to be augmented by asking each joining peer to report to $S(m)$ information concerning physical routes of its overlay edges.

D. Theoretical Result

We measure the performance of our algorithm by comparing its achievable throughput $f_{achievable}$ by the optimal value f^* obtained via the optimal algorithm introduced in Sec. II. More specifically, we define the *competitive ratio* of our algorithm as

$$\frac{f^*}{f_{achievable}}$$

and verify if there exists a worst case bound to the above ratio. We have the following result.

/* When Peer v_{new} Joins Session m^* /*	
Join (v_{new}, m, dem)	
1	for $j = 1$ to k
2	Join ($v_{new}, m, dem/k$)
/* When Peer v_{old} Leaves Session m^* /*	
Leave (v_{old}, m, dem)	
1	for $j = 1$ to k
2	Leave ($v_{old}, m, dem/k$)

TABLE II

MODIFIED ALGORITHM FOR THE CASE OF MULTI-PARENT STREAMING

Theorem 1: The online peer selection algorithm returns the competitive ratio bounded by $\log(|\mathcal{L}|)$.

E. Extension to Multi-Parent Streaming

The basic algorithm only allows each peer to select one parent. It can be easily upgraded to the case of multi-parent streaming, a highly effective strategy to improve overall throughput in many of today’s mesh-based and multi-tree-based P2P streaming systems[11], [8], [9]. The modified algorithm is shown in Tab. II, where each peer is allowed to have k parents.

The modified algorithm decomposes the multi-parent streaming problem as a collection of single-parent streaming problems for k sessions with the same set of peers. Each peer simultaneously belongs to k sessions, and independently chooses its parent in each session. Our algorithm does not require each peer to have k distinct parents. When the new peer v_{new} initially joins the session, it repeats the same action for k times, during each time it finds one parent whose distance to itself is the shortest, routes dem/k amount of traffic through the chosen overlay edge, and updating the lengths of corresponding physical links. This does not prevent the same peer from being chosen more than once as v_{new} ’s parent. In fact, it suggests that the overlay edge leading this peer to v_{new} has more available capacity, therefore should share more traffic than other candidate overlay edges.

From the claim made in Sec. III-D, it is straightforward that **Theorem 2** also applies to the case of multi-parent streaming.

IV. PERFORMANCE EVALUATIONS

A. Experimental Setup and Methodology

We use simulation to evaluate the performance of our algorithm. Two experimental topologies are chosen. The first one is a 100-node router-level network (200 edges) created by the Boston BRITE topology generator[12] using the Waxman model. Any pair of routers are connected by a pair of links with opposite directions. Peers are randomly attached to the routers in this network. The bandwidth of physical links between routers, as well as peers’ access links, are uniformly distributed from 10Mbps to 1024Mbps. The second topology follows the star configuration, where a central hub node, representing the Internet cloud, links to all peers. 1000 peers are created to connect to this hub. The bandwidths of their inbound and outbound bandwidths follow the same distribution of the first topology.

Over these two topologies, we simulate the lifetime of a P2P streaming session. For the star topology, each of the 1000 peers is scheduled to join and leave the session once. For the BRITE topology, a total of 100 peers joins and leaves the session.

B. Optimal Throughput

In Fig. 3, we first show the optimal throughput f^* calculated from the optimal algorithm introduced in [7].

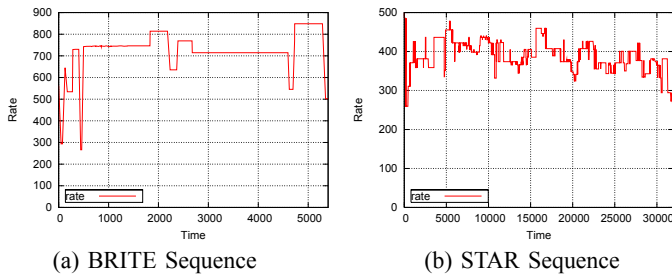


Fig. 3. Optimal Throughputs

We reemphasize that the above results are obtained by rearranging the peer streaming session from scratch at any time a peer joins or leaves. In the remainder of this section, we evaluate the performance of our online peer selection algorithm in terms of *throughput ratio*, the ratio of the throughput achieved by the online algorithm over the optimal throughput displayed in Fig. 3.

C. Performance of Online Peer Selection Algorithm

Fig. 4 shows the throughput ratios of the basic online peer selection algorithm. At the beginning and ending of both sequences, i.e., when the peer population is small, the online algorithm delivers optimal or near-optimal performance. The algorithm also consistently outperforms its theoretical lower bound $\log |\mathcal{L}|$ by a great extent⁴. Under the STAR sequence, the algorithm is able to maintain its throughput ratio at a stable level. Under the BRITE sequence, its performance fluctuates comparatively, and displays the general trend to perform less well as the peer population grows.

Also with multi-parent streaming, the algorithm performance doubles in most time points for BRITE sequence. However, sometimes 8-parent streaming delivers the best result, while at other points, the algorithm performs the best

⁴In BRITE topology, $|\mathcal{L}| = 200$, and $|\mathcal{L}| = 1000$ in STAR topology

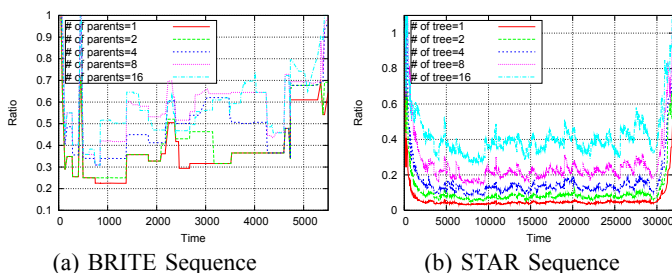


Fig. 4. Throughput Ratio of Multi-parent Streaming Algorithm ($\rho = 1$)

when number of parents is 16. Increasing the number of parents does not always lead to better performance, which indicates that the edge length function wrongfully reflect the traffic condition, a problem caused by overlooking the “link correlation”. In contrast, under the STAR sequence, allowing more parents is shown to consistently deliver better throughput ratio. With regard to “link correlation” problem, we note that both topologies we experiment with did not fully reveal the reality, but overstate and understate it respectively. Without public information on link capacity assignment in any real AS- or router-level setting, we choose the random distribution in BRITE topology.

On the other hand, the STAR topology presumably eliminates “link correlation”. What is clearly revealed here is the intolerance of our algorithm over “link correlation”. In reality, the performance of our algorithm is expected to be between its performances under these opposite scenarios.

V. CONCLUSIONS

This work aims to find practical peer selection solutions for maximum-throughput P2P streaming. The central challenge considered in this paper is peer churning, which is addressed by an online peer selection algorithm. Theoretically, we prove its approximation bound to the optimal throughput. Through evaluation under different topological setups and peer churning sequences, we show our solution to consistently deliver competitive throughput, which greatly outperform its theoretical bound.

REFERENCES

- [1] Michael Bishop, Sanjay Rao, and Kunwadee Sripanidkulchai, “Considering priority in overlay multicast protocols under heterogeneous environments,” in *Proc. of INFOCOM*, April 2006.
- [2] Yi Cui, Yuan Xue, and Klara Nahrstedt, “Maxmin overlay multicast: rate allocation and tree construction,” in *Twelfth IEEE International Workshop on Quality of Service (IwQoS)*, 2004.
- [3] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai, “Distributing streaming media content using cooperative networking,” in *ACM NOSSDAV*, New York, NY, USA, 2002.
- [4] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “Splitstream: High-bandwidth multicast in cooperative environments,” in *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.
- [5] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, “Bullet: High bandwidth data dissemination using an overlay mesh,” in *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
- [6] Yi Cui, Liang Dai, and Yuan Xue, “Optimizing p2p streaming throughput under peer churning,” in *Vanderbilt University Technical Report ISIS-07-811*, 2006.
- [7] Y. Cui, B. Li, and K. Nahrstedt, “On achieving optimized capacity utilization in application overlay networks with multiple competing sessions,” in *Proc. of ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2004.
- [8] “Pplive,” <http://pplive.com>.
- [9] “Uusee,” <http://uusee.com>.
- [10] V. Jacobson, *Pathchar*, <http://www.caida.org/tools/utilities/others/pathchar>.
- [11] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y. S. P. Yum, “Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming,” in *IEEE INFOCOM*, 2005.
- [12] A. Medina, A. Lakhina, I. Matta, and J. Byers, “Brite: An approach to universal topology generation,” in *Proc. of IEEE MASCOTS*, 2001.