ON OPTIMAL PEER-TO-PEER TOPOLOGY CONSTRUCTION WITH MAXIMUM PEER BANDWIDTH CONTRIBUTIONS

Tara Small, Baochun Li and Ben Liang

Department of Electrical and Computer Engineering
University of Toronto
{*tsmall, bli*}*@eecg.toronto.edu, liang@comm.toronto.edu*

ABSTRACT

As the number of participating peers scales up, multimedia streaming applications use amount of bandwidth from media streaming servers. As in previous works, we employ peer-to-peer (P2P) networks to mitigate unnecessary burden on the servers by using the upload bandwidth of peers to serve other peers. We formulate the network topology optimization problem as a minimization of server bandwidth cost, which leads to *scalability* of the system with respect to the number of peers participating in the session. We analytically design a topology that achieves this optimum, and a corresponding algorithm that generates it in practice. Using a simulation-based comparison study, we show that the optimization is achieved in a high-churn peer-to-peer network with realistic peer uplink capacities and link delays.

1. INTRODUCTION

Due to the high bandwidth demands of multimedia streams, deployment of multimedia services to support large numbers of users can be extremely expensive for the service provider. Peer-to-peer networks can be used in these scenarios because they rely not only on a small number of streaming servers to receive their data messages, but also on the uplink bandwidth of the participating peers themselves. Each participating peer contributes its uplink bandwidth to serve other peers, relieving the burden that would otherwise be imposed on dedicated servers. The most significant disadvantage of the peer-to-peer technology is that the peers are free to leave the network at any time; however, as the number of peers becomes large, the advantage of reduced server load clearly outweighs the disadvantage of transient peers.Peer-to-peer streaming offers compelling benefits to be implemented in real-world applications.

Although existing works in peer-to-peer streaming acknowledge the importance of scalability, they fail to offer sufficient insights towards the construction of peer-to-peer overlay topologies that *maximize* peer bandwidth contributions, and consequently minimize the load on dedicated servers. In this paper, we develop a rigorous analytical framework that minimizes server bandwidth cost in an environment where any pair of nodes have the potential to communicate.[1] By keeping a large set of potential connections available, we analyze an upper bound for the best case server bandwidth utilization (lowest server cost) in a peer-to-peer network for a given set of network characteristics. Based on our analytical insights, we design an algorithm to place new peers into the topology so that all peers contribute maximal bandwidth at any time and attain the objective of *scalability*. Furthermore, the optimized topology is resilient to churn with arbitrary distributions of peer lifetimes.

2. RELATED WORK

We categorize existing work towards topology construction for peer-to-peer streaming into two broad categories: *tree-based* and *gossip-based* topologies.

Tree-based topologies are rooted at the multimedia source, and the source manages all information for the construction and maintenance as peers are added and removed from the network (e.g. Coopnet [1], Splitstream [2]). Since all of the construction information is centralized, algorithms can potentially be very efficient; however, the removal of nodes close to the root of the tree can cause severe disruptions since a departure (or loss) of one peer affects all of its children. As a result, these topologies are vulnerable to high "churn" (departure) rates of peers. Maintenance algorithms for these topologies are complicated, do not scale well, and may not be effective in a high-churn environment. Due to the transient nature of participating peers in a streaming session, we believe that tree-based topologies are not particularly suitable for peer-to-peer streaming.

Mesh-based topologies, such as Bullet [3], are built on tree topologies, but disjoint sets of data are sent to different parts of the network so that less loss occurs when nodes leave the network, and information can be downloaded in parallel. The nodes in these topologies receive some of the media from their parents in the underlying tree, but are responsible for finding the remaining packets themselves. Any parent node

[1]This assumption is practical in a P2P network that is streaming over the Internet.

will attempt to serve its peers with as much spatial diversity as possible. We concur that mesh-based topologies are superior with respect to the total available uplink capacities, though they are still quite rigid, as they are based on trees.

Gossip-based or data-driven protocols, such as CoolStreaming [4], GridMedia [5], and Chunkyspread [6], have been proposed to "spread" data to M *randomly* chosen neighbors, using either "push" or "pull" techniques. Each data message received by a peer is forwarded to a random set of other peer nodes. The uniform random choice of peers to serve has been proven robust to dynamic changes (churn); however, peer utilization only improves using limited local information. Gossiping strategies do not choose neighbors to globally optimize any particular metric, such as bandwidth costs at the servers.

Unlike some of the previous works (e.g. [7]), we firmly believe that peers have *asymmetric* uplink and downlink bandwidth capacities (with considerably smaller uplink capacities), as they are mostly served by ADSL or cable broadband connections. In particular for very high-bandwidth applications, nodes may receive data from multiple upstream peers, and the gap between the total available download and upload capacities in the streaming session should be bridged by dedicated streaming servers.[2] Alternatively for lower quality streaming where the playback rate is lower than the uplink capacities of the peers, it is possible for a single peer to serve another peer.[3] Our objective is to minimize the bandwidth required from the dedicated multimedia servers.

3. PEER CONTRIBUTIONS IN OVERLAY TOPOLOGIES

Let us assume that the peer-to-peer network considered here consists of *one* multimedia source (known as "the server") and $N-1$ *peers*. Note that having multiple streaming servers in the network is functionally equivalent to having one server with the same total uplink bandwidth. The dedicated streaming server continuously generates data messages that form the multimedia stream, to be served to all peer nodes in the network. Peers can relay the stream at a random upload rate U, and the media must be played back at a fixed rate p. Whenever possible, each peer receives the media stream forwarded from (possibly several of) their peers. If the serving peers are unable to jointly send messages at the bit rate p, either because they have not yet received the packets required by their neighbors or because they have used all of their uploading bandwidth, the remaining unserved rate is served from the dedicated server. Hence, the choice of topology has a significant effect on the bandwidth cost to the server.

Tree-based peer-to-peer topologies are constructed at the multimedia source in a centralized manner, with maintenance algorithms to keep the global server cost low. Gossip-based topologies require each peer, including the server, to choose M random peers as "partners" that could potentially serve the media stream, so the network is robust to churn; however, the topology is not globally optimized. We wish to take the positive aspects of these different topologies to construct the best possible topologies to minimize server bandwidth costs without causing unreasonable delays for reception at the peers. For example, if p is greater than the peer upload bandwidth U then there are opportunities for each peer to be served by multiple other peers in the session, while using the entire upload bandwidth of those peers. We construct a topology that uses all upload bandwidth for all peers.

First consider a peer l, the peer that receives the data stream at *the latest* time. The upload bandwidth at peer l is necessarily idle because all other peers would already have received the data. We choose a set of peers to serve l such that the combined upload capacity of the peers is as close as possible to p (but not greater). These peers are referred to as the $(l-1)$-peers. In a similar fashion, we assign a collection of peers to serve each of the $(l-1)$-peers; however, we allow some of the excess peer upload bandwidth to serve the small unserved portion of peer l. This process is repeated until all peers are filled into positions in the topology, and any remaining peer that is not served by other peers is served by the server. We refer to the set of peers served by the server as 1-peers, the peers they serve as 2-peers, ...

The server bandwidth cost in a peer-to-peer topology is the difference between the bandwidth required by the peers to stream the data messages and the uplink bandwidth contributed by the peers. Therefore since, by construction, this topology uses the most possible upload bandwidth from the peers, it imposes minimal server bandwidth cost. To achieve this topology in a dynamic environment, we add peers according to *Algorithm 1* when they request insertion into the network.

Algorithm 1. Immediately before a new peer joins the streaming session, let the bandwidth difference between the k- and $(k+1)$-peers be

$$\delta_k := \begin{cases} (\sum_{j\in k\text{-peers}} e_j) - (|(k+1)\text{-peers}| * p \\ \qquad - \sum_{m\in(k+1)\text{-peers}} s_m) & \text{for } 1 \le k < l \\ \sum_{j\in l\text{-peers}} u_j & \text{for } k = l, \end{cases}$$

where p is the playback rate of the media, u_j is the total upload bandwidth of peer j, e_j is the unused (excess) upload bandwidth of peer j, and s_m is the bandwidth served to peer m from other peers. Let $k^* = \max_k \arg\{|\delta_k|\}$. If $\delta_{k^*} < 0$, the new peer is inserted as a k^*-peer, which can potentially be served by any (k^*-1)-peer and to help serve any (k^*+1)-peers. If $\delta_{k^*} > 0$, the new peer is inserted as a (k^*+1)-peer, to be served by the k^*-peers and to serve the (k^*+2)-peers.

Algorithm 1 places peers in a directed graph such that the server cost is increased as little as possible each time a new peer is inserted by using as much bandwidth as possible from

[2] Such views are shared by PROMISE [8] and PALS [9].

[3] This perspective is suggested by CoolStreaming [4].

the peers in the network. Although this is a centralized algorithm, new nodes could be placed in a distributed network if the peers could approximate δ_k using local information. Since any peer that has idle upload bandwidth can potentially serve any new peer at any time, the server cost is not only minimized in a local sense, but also in a global sense because the placement of a peer at an early stage will never hamper peer placements at later stages.

This topology construction employs structure in the topology, as for tree-based topologies, to strive for global optimality of server cost. At the same time, it can borrow the robustness of the gossiping strategies to choose the serving peers, by allowing each $(k+1)$-node to sample from any of the k-nodes at random to serve it. We have incorporated both global optimality and resilience to the churn of a peer-to-peer network.

4. PERFORMANCE EVALUATION

We compare the performance of the optimized topology to the topologies from previous works in the presence of churn using a simulation-based study. In particular, we consider a bandwidth-optimized tree topology, with four peers connected directly to the server and the other peers forming chains down from those four. We also compare to a binary tree. Though tree topologies typically react poorly to churn, we wish to compare the wasted upload bandwidth of the peers in the topology rather than their resilience to churn. Therefore, we assume that the trees can be immediately reconstructed if any peer leaves.

Gossiping-based topologies are also evaluated in the comparison study. Each peer connects to M random neighbors (serving peers) and, again since we want to compare the peer bandwidth contributions rather than the recovery mechanism, each peer finds a new partner immediately if one of its neighbors goes offline. We assume that the our proposed scheme constructs its topology as described in Algorithm 1.

The results of our evaluation are shown in Fig. 1. In this set of tests, we assume each link has normalized length 1, play rate $p = 225$ [kbps], and an upload capacity is chosen from a Zipf distribution with mean $u = 100$ [kbps].[4] The lifetimes of the peers are also chosen from a Zipf distribution,[5] where there is high probability of smaller lifetimes and low probability of long lifetimes. We truncate the Zipf distribution so that the mean is 270 time-steps (45 minutes). The insertion process is Poisson, with rate oscillating slowly between 6 peers/time-step and 0.25 peers/time-step, for 10-second time-steps to represent times when there are more and fewer peers online.[6] Note that the connections of points in the curves indicate points adjacent in time in the simulation. When the rate of insertion of nodes in the network is large, we see points closer to the right side of the curves. When the rate of insertion is lower, then more nodes are expiring than are being introduced; so we see the points of the curve moving toward the left. Since there is no preference for which nodes expire, the topology may become less optimal. We see from the Fig. 1 that Algorithm 1 is able to insert new nodes without getting trapped in local minima, so the network recovers well and achieves stability even though there is churn in the network. We assess three metrics: (1) the average distance from a peer to the streaming server; (2) the serving bit rate from the server; and (3) the total idle upload bandwidth at the peers.

Fig. 1(a) compares the average distance to the server (data delay) as a function of the number of peers in the system, which varies due to the rates of insertion and removal. Since $U < p$, the proposed peer-bandwidth-optimized topology keeps many peers close to the server, and the time from transmission from the server to reception at the peers is very short on average. Normally, one would expect that lowering the delay of packet reception would increase the cost to the media server; however Fig. 1(b) shows that the opposite is true, even with conservative assumptions for recovery in the gossip-based and the tree topologies! Our topology achieves lower delay than either the binary tree or the gossip-based topologies, while using less server bandwidth. On the other hand, the server cost for the centralized tree with four chains achieves similar cost since the peer uplink is also almost completely utilized, as shown in Fig. 1(c). However, we assumed that the tree topology with four chains would be able to immediately adapt to changing topologies, which is not true in practice. The bandwidth-optimized topology reacts to the change and its bandwidth may oscillate while it samples from the remaining nodes that could potentially serve, but this topology is practical and has also shown considerably less delay.

5. CONCLUSION

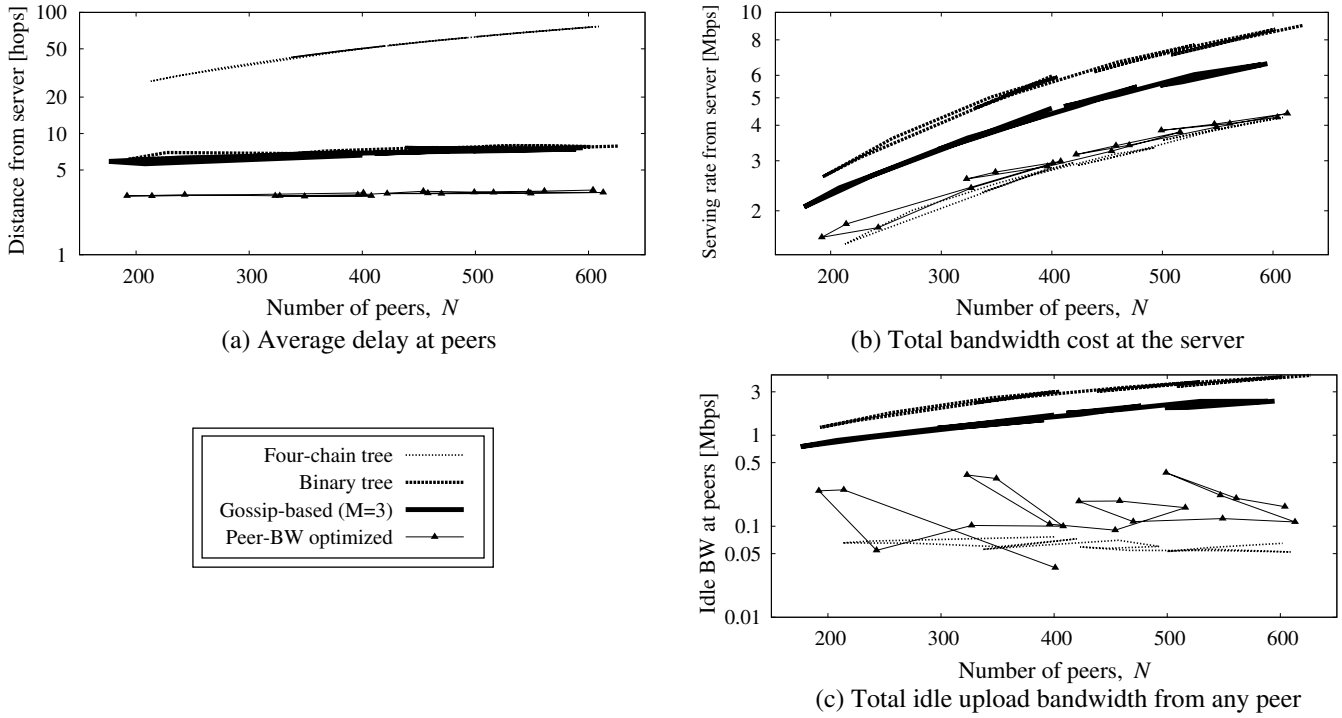
In this paper, we have constructed a network topology that maximizes peer-to-peer uplink bandwidth resources and facilitates scalability in multimedia streaming. This topology adapts and loosens the idea of structure from tree topologies that perform poorly in high-churn environments, but is able to achieve global optimization. By keeping a small amount of structure, namely the number of hops from the source, and also incorporating gossiping to select neighbors from a subset of peers, we are able globally minimize the cost to the server and also be resilient to churn.

Our optimized topology is based on an analytical minimization for an idealistic model, but we have shown that it is also flexible. This topology achieves better performance than previous topologies in systems where peers have realistic dis-

[4]By using this distribution, we are attempting to take into account that peers have different inherent upload capacities and that the peers are likely multitasking and using some of their uplink bandwidth for other purposes.

[5]The Zipf distribution has been shown to represent the online lifetimes of human users [10].

[6]For example, there may be more peers streaming multimedia in the evenings or on the weekends.



(a) Average delay at peers

(b) Total bandwidth cost at the server

(c) Total idle upload bandwidth from any peer

Fig. 1. Performance metrics using different peer-to-peer topologies with peer uplink bandwidth capacities and lifetimes following Zipf distributions.

tributions for peer lifetimes and uplink capacities, and that the uplink capacities may be lower than the playback bit rate. While encouraging peers to utilize their uplink bandwidth as completely as possible, this topology allows applications to scale to larger network sizes that may previously have overwhelmed a multimedia source and at the same time achieves low latencies.

6. REFERENCES

[1] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 2002.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," in *Proc. 2nd International Workshop on Peer-to-Peer Systems*, February 2003.

[3] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proc. 19th ACM Symposium on Operating Systems Principles*, October 2003.

[4] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," in *Proc. INFOCOM 2005*, March 2005.

[5] M. Zhang, L. Zhao, J. Luo Y. Tang, and S. Yang, "GridMedia: A Peer-to-Peer Network for Streaming Multicast Through the Internet," in *Proc. ACM Multimedia 2005*, November 2005.

[6] V. Venkatraman and P. Francis, "ChunkySpread Overlay Multicast," in *Proc. 2nd Symposium on Networked Systems Design and Implementation*, May 2005.

[7] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Proc. Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[8] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," in *Proc. ACM Multimedia 2003*, November 2003.

[9] R. Rejaie and A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming," in *Proc. 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, June 2003.

[10] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," in *Proc. SIGCOMM'04*, September 2004.