UN-LEECHING P2P STREAMING BY ACTIVE OVERLAY MANAGEMENT

Jeonghun Noh, Pierpaolo Baccichet, Aditya Mavlankar, and Bernd Girod

Information Systems Laboratory, Department of Electrical Engineering
Stanford University, Stanford CA 94305, USA
Email: {jhnoh, bacci, maditya, bgirod}@stanford.edu

ABSTRACT

We propose peer swapping as a means to improve the throughput of structured-overlay-based peer-to-peer (P2P) streaming systems. In many P2P streaming systems, an overlay is incrementally constructed as peers join the system. When the system is dominated by peers with low uplink bandwidth (so-called leeches), the overlay can get clogged, which hampers its growth. A P2P system that suffers from clogging cannot scale any longer because even peers with high bandwidth are rejected. We first provide a theoretical analysis of clogging, which reveals that clogging is inevitable in a structured-overlay-based streaming system with heterogeneous peer uplink bandwidths. To overcome clogging, we propose distributed peer swapping, a method to reconfigure the overlay without centralized control. Experimental results demonstrate that peer swapping enables the system to accept more peers by mitigating clogging.

1. INTRODUCTION

Many P2P video streaming systems [1, 2, 3] use a tree-type overlay for efficient media distribution. In this approach, one or multiple spanning trees are constructed to connect all the peers participating in a streaming session. The video source is the root of each tree and peers potentially offer to forward, fully or partially, the video traffic to other peers. In these structured overlays, peers without relaying capabilities, sometimes called *leeches*, may prevent new peers from connecting to the system. This phenomenon is called *clogging*. In this paper, we propose peer swapping as a means to overcome clogging. Peer swapping allows peers to actively exchange their positions in the tree overlay so that a high bandwidth peer can take over the position of a leech. After the high bandwidth peer becomes part of the system, it receives and relays data to the leech it swapped with.

Yi Cui et. al. considered the max-min resource allocation among peers in [4]. The same authors in [5] investigated how to maximize the aggregate utility of all receivers, where a peer's utility is defined as a function of its streaming rate. The basic assumption they applied to these studies was that peers benefit from higher downlink streaming rates. However, we consider a streaming system where peers exhibit identical satisfaction as long as they receive video at the full bit rate, because the video is encoded at a single bit rate. MutualCast [6] achieves the full utilization of peers' heterogeneous uplink bandwidth. However, with MutualCast, peers allocate smaller uplink bandwidth per child peer as more peers join the system, increasing end-to-end delay. In this study, we consider a system with a fixed number of multiple trees, where the uplink bandwidth a peer allocates per child is fixed. Such a system can avoid excessively increasing end-to-end delay with more peers joining the system, and improve its average throughput with the assistance of peer swapping.

The remainder of the paper is organized as follows. Section 2 presents a combinatorial analysis of clogging when the overlay is not managed actively. In Section 3, we extend our previously proposed P2P streaming system based on multiple multicast trees [7] by incorporating the peer swapping method as a distributed protocol. Experimental results are presented in Section 4.

2. SYSTEM ANALYSIS

2.1. Preliminaries

We assume that an overlay structure consisting of complementary multicast trees is incrementally constructed for data distribution as peers join a system. The video is encoded at a constant bit rate R, which remains identical throughout the entire video streaming session. Downlink bandwidths of peers are assumed to be larger than R, thus every peer can receive the video stream at the full rate. However, the uplink bandwidths of some peers are assumed to be smaller than R, which reflects the asynchronous uplink and downlink of today's common access network technologies.

We define a *leech* as a peer that does not contribute to a system or contributes with a limited uplink bandwidth, but that nevertheless receives the video at the full bit rate R. For instance, in a practical system, peers behind firewalls or some NATs may not be able to forward data to other peers. When leeches block all the data paths in the overlay, no new peers

This work has been supported, in part, by the Max Planck Center for Visual Computing and Communication.

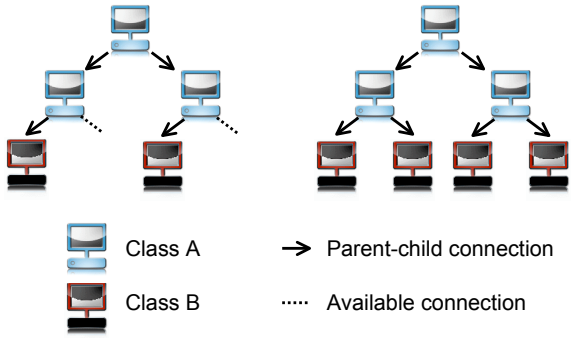

Fig. 1. Left: A system with available bandwidth. Right: A system experiencing clogging. All the paths from the tree root are blocked by Class B peers.

can join the system. A P2P system is said to suffer clogging when it cannot accept any more peers.

2.2. Probability of System Clogging

We define the capacity of a tree-overlay-based P2P system as the aggregate of the uplink bandwidths of peers that intend to join the system including the source peer. Whereas, the system throughput is defined as the aggregate of the uplink bandwidth of peers connected to the overlay. To understand how clogging affects the system throughput, we consider a simplified case where the system with a single tree accommodates two classes of heterogeneous peers. Suppose that $\frac{N}{2}$ Class A peers have an uplink bandwidth of $2R$, and $\frac{N}{2}$ Class B peers have an uplink bandwidth smaller than R. Class A peers can have up to two child peers each, whereas Class B peers cannot have any child peers as a bandwidth of R is required to support one child peer. When a peer from either class joins the system, we assume that the peer can find a parent peer with available bandwidth, if one exists. In Figure 1, instances when the system can and cannot accommodate new peers are contrasted.

In Figure 2, peer joins are modeled as a random walk. The horizontal axis shows a peer join sequence in discrete-time according to the order of their arrival times. The vertical axis represents the system's total bandwidth available for new peers. The unit of the vertical axis is the number of new peers that can be accepted. The random walk starts at $(0, B_0)$ where B_0 denotes the source peer's bandwidth. An up-step occurs when a class A peer arrives; the class A peer consumes one unit of the system resource, while providing two units. Thus, a class A peer entails a net gain of one unit. A down-step occurs when a class B peer arrives; it consumes one unit. The random walk finishes at (N, B_0) because the net contribution of N peers is 0. System clogging occurs when the random walk hits the horizontal axis.

We apply the *Ballot Theorem* to this problem to compute the likelihood of system clogging. The original Ballot Problem is as follows [8]: Suppose that in an election, candidate A receives a votes, and candidate B receives b votes, where $a \geq kb$ for some positive integer k. In the Ballot Problem, one computes the number of ways the ballots can be ordered so that A maintains more than k times as many votes as B throughout the counting of the ballots.

For our problem, k is set to 1 and $a = b = \frac{N}{2}$. Unlike the original Ballot Problem, we set the initial point as B_0. We compute the percentage of paths that stay above the horizontal axis among all of the possible paths; these are called *good paths* (See Figure 2). Instead of counting good paths, however, we choose to count *bad paths* using the "reflection method" [8]. In the reflection method, the first down-step that meets the horizontal axis is identified. Then, the initial portion of the path, between the starting (leftmost) point and the identified point, is reflected across the horizontal axis. As illustrated in Figure 2, every bad path has a reflection path which starts at $(0, -B_0)$ and ends at (N, B_0). Moreover, every path from $(0, -B_0)$ to (N, B_0) has a corresponding path from $(0, B_0)$ to (N, B_0) that touches or crosses the horizontal axis at least once. Based on this one-to-one mapping relationship, we conclude that the total number of bad paths is the same as that of all paths from $(0, -B_0)$ to (N, B_0). There are $(\frac{N}{2} + B_0)$ up-steps and $(\frac{N}{2} - B_0)$ down-steps in each bad path. Assuming $N > 2B_0$, the number of all bad paths is obtained as $\binom{N}{\frac{N}{2}+B_0}$.

The probability of system clogging is represented as the ratio between the number of bad paths and the number of total paths, represented in this equation

$$\begin{aligned} \Pr(\text{system clogging}) &= \frac{\text{Number of bad paths}}{\text{Number of total paths}} \\ &= \frac{\binom{N}{\frac{N}{2}+B_0}}{\binom{N}{\frac{N}{2}}} \\ &= \prod_{i=1}^{B_0} \frac{\frac{N}{2}-i+1}{\frac{N}{2}+i}. \end{aligned} \quad (1)$$

Figure 3 shows the probability of system clogging depending on the bandwidth of the source (B_0) and the number of peers (N). From the figure along with our combinatorial analysis of the example above, we draw the following conclusions. (1) The probability of clogging decreases when B_0 increases. (2) Clogging is more probable when the system size N increases, even when B_0 is high. (3) Peer join sequences have a significant impact on the actual system throughput. From our findings, we conjecture that clogging is inevitable in P2P systems serving peers with more heterogeneous uplink bandwidths than bi-class peers.

3. IMPROVING THE P2P SYSTEM THROUGHPUT

In the previous section, we showed that clogging may occur when the overlay is primarily shaped by the peer join sequence. This section introduces a distributed protocol that

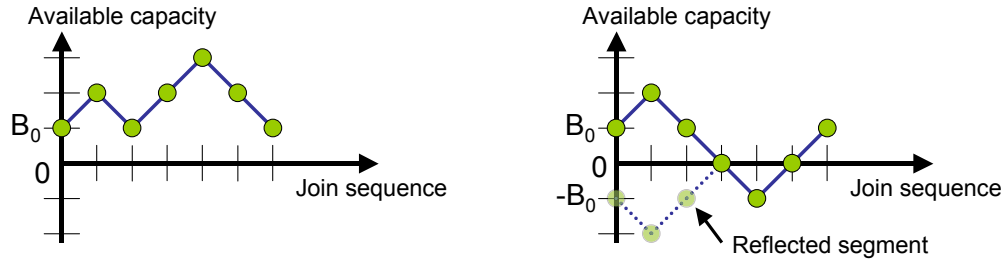

Fig. 2. A random walk model for total available uplink bandwidth in the overlay. B_0 is the number of peers that the source peer can directly serve. Left: Example of good path. Right: Example of bad path. The reflected segment starting from $-B_0$ is depicted with a dotted line.

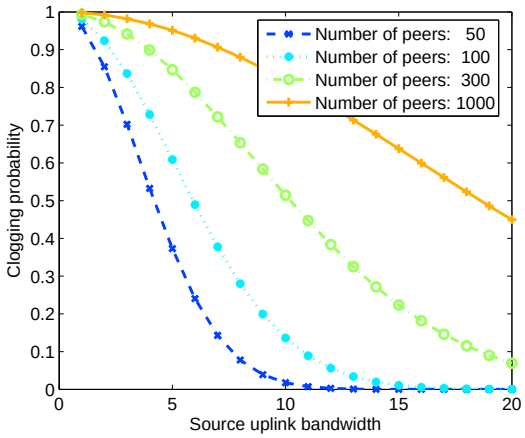

Fig. 3. Probability of system clogging under incremental overlay construction. Source uplink bandwidth is the number of child peers the source peer can directly serve.

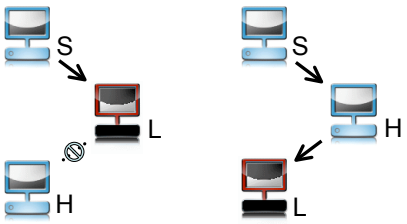

Fig. 4. (a) Bad join sequence (b) Good join sequence

employs peer swapping as a means to allow the system to overcome the disadvantage due to unfavorable peer join sequences.

3.1. Peer Swapping

Suppose that S is the video source, delivering video of bitrate R to a P2P system. Let us assume that Peer L and Peer H have $R_L < R$ and $R_H \geq R$, respectively, where R_i is the uplink bandwidth of Peer i. Suppose that H wants to join the overlay in Figure 4 (a). Since L is clogging the data path, H cannot join the system. However, in Figure 4 (b), L can join the overlay by connecting to H. Peer swapping transforms the overlay from Figure 4 (a) to (b), so that the system can scale by overcoming clogging.

3.2. Distributed Protocol for Peer Swapping

New peers can detect and resolve clogging without the help of any central server. Suppose that a new peer C contacts the source S to join a P2P multicast. Source S responds with a list of randomly drawn peers that might potentially serve as parents. By probing these peers, C collects partial information about the P2P overlay. If C finds a peer with available bandwidth, it connects to the peer and starts to receive a substream.

When C fails to find any peer with available bandwidth, it concludes that the system is clogged. Upon the discovery of clogging, C performs self-admission control by comparing its own uplink bandwidth, R_C, with R_{thres}. A threshold R_{thres}, set equal to R, is used to determine an incoming peer is a leech. When R_C is below R_{thres}, it classifies itself as a leech and is not allowed to join the system. If R_C exceeds R_{thres}, then C selects the leech with the shortest path to the source among those that have responded to the probing. Let B denote the selected leech. Then, C contacts B's parent node, denoted as A, in order to request peer swapping between B and C. Once the swapping is completed, A becomes C's parent and C becomes the new parent of B. After C becomes part of the network, the system throughput increases by C's uplink bandwidth R_C. If no leeches are found during the probe step, C repeats the procedure by obtaining a new list of potential parents from the source S until it finds a leech to swap with.

3.3. False Detection of System Clogging

Since the detection of system clogging is performed without any central coordination, a false detection is possible. A false detection of clogging occurs when none of the available peers

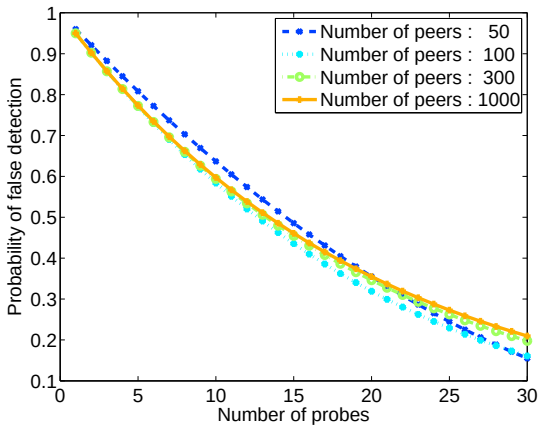

Fig. 5. Probability of false clogging detection with a different number of potential parents (probes). 5% of peers in the system have available bandwidth at the time of probing.

Run	1 Tree		2 Trees		4 Trees	
	NPS	PS	NPS	PS	NPS	PS
1	4	52	100	100	100	100
2	8	52	100	100	3	100
3	5	52	3	100	100	100
4	2	52	2	100	100	100
5	2	52	5	100	100	100
6	10	52	100	100	6	100
7	9	52	100	100	100	100
8	7	52	100	100	5	100
9	13	52	100	100	100	100
10	2	52	4	100	100	100
11	8	52	2	100	100	100
12	2	52	100	100	100	100
Average	6	52	59.67	100	76.17	100

Table 1. The percentage of peers connected to all trees. Each row is obtained with the same peer join sequence. (NPS: no peer swapping, PS: peer swapping)

in the system are discovered through probing. This can happen because probing random peers provides only partial information about the entire system. The consequences of false detection are insignificant for peers that are not leeches. However, in instances of false detection, leeches are not allowed to join the system. We analyzed the impact of false detection for leeches. Suppose that p fraction of N_p peers are available for new peers at a given time. When a new peer joins, the peer randomly probes r peers out of N_p by sending probe packets. The probability that the peer finds no parent among the available peers is expressed as $\frac{\binom{(1-p)N_p}{r}}{\binom{N_p}{r}}$.

In Fig. 5, the rate of false detection is equal to the rejection rate of leeches, with 5% of peers in the system having available bandwidth at the time of probing. Regardless of the group size, more probes decrease rejection rate. For example, when the group size is reasonably large (>300), probing 15 random peers results in 45% of false detection. If additional 15 random peers are probed, the false detection rate drops to 20%. While the false detection rate does not drop to 0%, it is still possible, most of the time, for leeches to discover available peers by probing a small portion of the peers in the system.

4. EXPERIMENTAL RESULTS

We integrated the peer-swapping algorithm presented in Section 3 into our implementation of the SPPM (Stanford P2P Multicast) video streaming system [7, 9] within the NS-2 network simulator. The 10-second long video sequence of *Foreman*, encoded with H.264 at the rate of $R = 670$ kbps, is delivered from the source peer. The uplink bandwidth of the source peer is applied to $2R = 1340$ kbps. 50 Class A peers with $1.5R$ of uplink bandwidth, and 50 Class B peers with $0.5R$ of uplink bandwidth are placed on the randomly chosen edge nodes of the backbone network. Peers join the system at an arbitrary time, independently of each other. They remain in the system until the video session is over. A flash crowd period, during which peers arrive faster than the average arrival rate, is avoided to study the effect of the incremental construction of trees with sequential peer joins.

Table 1 shows the experimental results with randomly generated peer join sequences. We selected twelve representative runs to allow for one-on-one performance comparison. The first column indicates the index of a peer join sequence. The following columns correspond to the results for 1, 2, and 4 complementary trees, respectively. For each simulation run, we counted peers connected to all trees. For each case, experiments with and without peer swapping were performed. The results demonstrate that the system throughput varies depending on the peer join sequence without peer swapping. However, a system with an actively managed overlay manages to connect all peers, except for a system with a single tree. In the system with one tree, the system throughput is $52R$, which is the sum of the aggregate of the uplink bandwidths of all the Class A peers, plus the source uplink bandwidth. This limitation arises because Class B peers cannot contribute their resources as their uplink bandwidth is below the tree bit rate R. This limitation is resolved with the 2 and 4 trees because in these instances, class B peers can fully contribute their resources to the system.

We now turn to the case where peers leave the system independently of each other. For peer arrival and departure, peers' on-time and off-time are assumed to be exponentially distributed with 90 seconds and 10 seconds on average, respectively. Each simulation has a length of 600 seconds. When peers leave the system, their child peers suffer disconnection. The disconnected peers then find new parents to

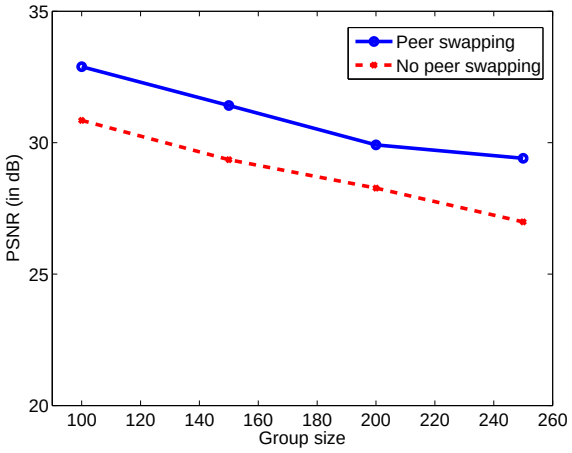

Fig. 6. Average video quality of individual peers for systems with and without peer swapping. 4 complementary trees are built in the overlay.

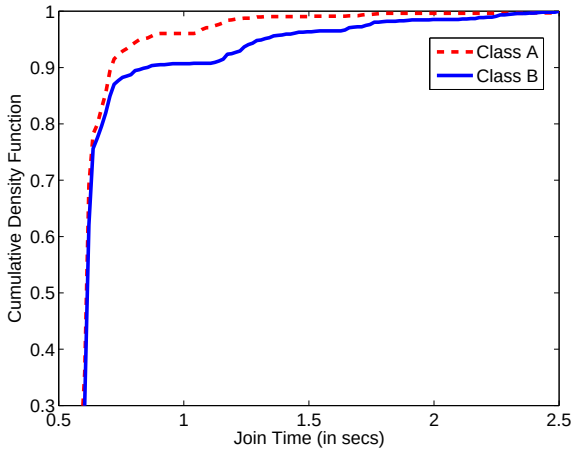

Fig. 7. Cumulative density function of join time. 4 complementary trees are built in the overlay. Class B peers experience longer join time than Class A peers.

continue to receive the video stream. During the recovery, the system may experience clogging depending on the reconnection order of peers similar to peer's join process. Figure 6 shows the average video quality in PSNR for the system with 4 complementary trees. Each point on the curves was obtained by averaging the PSNR of peers from 10 simulations. A gain of more than 2 dB was observed for a system with an actively managed overlay compared to a system without such measures. Due to longer tree depth and more disruption in video transmission, the video quality decreases as the group size (number of peers) increases both with and without the proposed peer swapping.

Figure 7 illustrates the CDF of a peer's join time. Peers are allowed to join and leave the system arbitrarily. Class B peers (leeches) require more time to join the system than Class A peers, because Class B peers are rejected when a system has no available bandwidth. Once a new Class A peer joins the system via peer swapping, the rejected Class B peers can join the system, thus resulting in longer join time.

5. CONCLUSION

In this paper, we propose the use of peer swapping, as a means to improve the throughput of P2P streaming systems. Peer swapping enables the system to accept more peers by swapping the positions of peers in the overlay during the streaming session. Our analysis of the probability of clogging reveals that clogging is inevitable when the overlay is constructed incrementally as heterogeneous peers join the system. Simulation results show that without peer swapping, almost 25% of the peers are rejected by the system with four multicast trees. However, peer swapping allows all the peers to be served irrespective of their arrival order.

6. REFERENCES

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *Proceedings ACM SIGCOMM*, Aug. 2002.

[2] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," *Proc. ACM NOSSDAV, Miami Beach, FL*, May 2002.

[3] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," *Proc. IPTPS'03, Berkeley, CA*, Feb. 2003.

[4] Y. Cui, Y. Xue, and K. Nahrstedt, "Max-min Overlay Multicast: Rate Allocation and Tree Construction," *IEEE International Workshop on Quality of Service (IwQoS), Montreal, Canada*, June 2004.

[5] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal Resource Allocation in Overlay Multicast," *IEEE Transactions on Parallel and Distributed Systems*, 2006.

[6] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: An efficient mechanism for one-to-many content distribution," *ACM SIGCOMM ASIA Workshop*, Apr. 2005.

[7] E. Setton, J. Noh, and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," *Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia, Singapore*, pp. 39–48, Nov. 2005.

[8] M. Renault, "Four Proofs of the Ballot Theorem," *http://webspace.ship.edu/msrenault/ballotproblem*, Aug. 2007.

[9] E. Setton, P. Baccichet, and B. Girod, "Peer-to-Peer Live Multicast: A Video Perspective," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 25–38, Jan. 2008.