

Considering Altruism in Peer-to-Peer Internet Streaming Broadcast *

Yang-hua Chu
Carnegie Mellon University
yhchu@cs.cmu.edu

Hui Zhang
Carnegie Mellon University
hzhang@cs.cmu.edu

ABSTRACT

In peer-to-peer overlay for video broadcast, peers contribute a portion of the bandwidth to the overlay in return for the service. In the presence of network heterogeneity, it is not well understood how much bandwidth peers should contribute and receive in return. Existing protocols implicitly assume peers are either completely altruistic (which leads to fairness concerns) or completely selfish (which leads to sub-optimal performance). In this paper, we argue that altruism should be explicitly considered. We propose a policy framework in which a wide range of altruism can be modeled and parameterized. The key findings are (i) the level of altruism has significant implication on the overall performance of the receivers; even a small degree of altruism goes a long way in improving their performance, and (ii) a wide range of altruism policy can be implemented efficiently in a distributed fashion. We validate these claims using simulation, with traces from real Internet broadcast events.

Categories and Subject Descriptors: C.2 [Computer Systems Organization]: Computer Communication Networks

General Terms: Performance, Design

Keywords: Peer-to-peer, Overlay Multicast, Video Streaming, Altruism

1. INTRODUCTION

Due to the scaling and deployment concerns with IP Multicast, peer-to-peer overlay multicast is poised as a promising alternative to support multicast applications over the

*This research was sponsored by DARPA under contract number F30602-99-1-0518, and by NSF under grant numbers Career Award NCR-9624979 ANI-9730105, ITR Award ANI-0085920, and ANI-9814929. Additional support was provided by Sloan Research Fellowship and Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, or the U.S. government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NOSSDAV'04, June 18–18, 2004, Cork, Ireland.

Copyright 2004 ACM 1-58113-801-6/04/0006 ...\$5.00.

Internet. In this paper, we are particularly concerned with *altruism*, an inherit prerequisite for peer-to-peer networks which has largely been overlooked by overlay multicast research. We study altruism in the context of video streaming broadcast, which has a stringent requirement to deliver high bandwidth data in real time.

Altruism is a key element of peer-to-peer overlay multicast. Consider a local TV station that wants to broadcast daily news over the Internet. In the absence of IP Multicast, the sender (TV station) must replicate data individually to each receiver. This requires costly bandwidth provisioning at the sender. Overlay multicast addresses this problem by utilizing the forwarding capacity at the receivers. Thus the sender only needs to replicate the data to a few receivers, and the rest of the forwarding/replication task is shared among the receivers. For overlay multicast to succeed, the premise is that the receivers (peers) are willing to contribute (some of) their forwarding bandwidth to the overlay, which is an altruistic behavior that may not happen.

Existing protocols implicitly assume peers are either completely altruistic or completely selfish, and tailor their designs to the specific altruism assumptions. Bullet [4] and ESM [3] assume peers are completely altruistic in contributing all of their forwarding bandwidth. This leads to a socially optimal solution which maximizes the receiving bandwidth to all peers. However, with the prevalence of free-riding [1], it is not clear whether peers would always behave with complete altruism. CoopNet [6] and SplitStream [2] have taken the opposite extreme by implementing a *Bit-for-Bit* policy. In Bit-for-Bit, peers contribute only as much bandwidth as they receive. Though Bit-for-Bit seems more “fair”, the resulting performance is far from socially optimal. In particular, peers behind asymmetrical links (such as ADSL) cannot contribute much bandwidth, and therefore receive poor video quality (or no video) in return.

In this paper, we *explicitly* consider altruism as a parameter. Streaming applications can tune this parameter to trade performance with fairness. The rest of the paper then addresses the following two questions: (i) how to parameterize altruism in an overlay multicast protocol? and (ii) is it feasible to design a flexible protocol that can implement a wide range of parameterized altruism policies?

We begin by designing a framework that captures a class of policies with varying levels of altruism (Section 3). The altruism of a peer is expressed as the ratio (K) of the bandwidth contributed to the overlay and the bandwidth received in return, where K is between 1 (Bit-for-Bit) and ∞ (complete altruism). For peers with high forwarding capacity, K

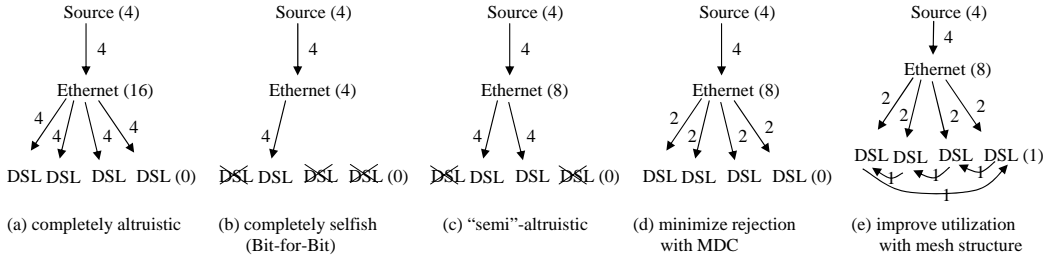


Figure 1: Examples to illustrate the effect of altruism on the contributing and received bandwidth of peers. (d) and (e) show two optimizations used in the paper. Each unit bandwidth is 100Kbps.

bounds the extra bandwidth donated to the overlay. The extra bandwidth is evenly distributed among peers with low forwarding capacity.

Next we show such a framework of policies can be implemented in a distributed overlay multicast protocol (Section 4). To implement the policy in finer bandwidth granularity, we need a scalable codec and an efficient overlay structure. We leverage Multiple Description Codec (MDC) and take ideas from protocols that construct a multiple disjoint trees structure [6, 2]. The key challenge is that bandwidth allocation requires global knowledge, i.e. the “fair” receiving bandwidth of a peer depends on the bandwidth capacity of all other peers. To implement specific altruism policies, we regulate the number of stripes a peer should receive by assigning a *priority* level to each peer when it joins a tree. Peers with higher priority in a tree can *preempt* other peers with lower priority.

We evaluate the importance and feasibility of parameterized altruism in simulation, with traces from real Internet broadcast events (Section 5). Our results indicate that there is a significant performance benefit with just a small level of altruism, and this benefit is more pronounced in a more heterogeneous peer environment. Moreover, the proposed distributed protocol can efficiently implement a wide range of altruism policies that is comparable to a centralized protocol with global knowledge.

2. BACKGROUND

In this section, we begin with an example that illustrates the impact of altruism on peer performance with a single tree overlay structure. Our example also indicates the limitation of a single tree approach in supporting a flexible range of altruism policies. We then describe two basic building blocks proposed in [6, 2] that we leverage in this paper: multiple tree structure and MDC.

2.1 Motivating Example

Consider Figure 1 which depicts an example overlay multicast tree with a single source and 5 receivers. The source broadcasts a video stream at a rate of 400Kbps with no special encoding (no MDC). The maximum forwarding capacity of the source is 400Kbps. The receivers are behind either a DSL or Ethernet connection. Peers behind DSL can receive the video stream but not forward one. The peer behind Ethernet has at least 1600 Kbps forwarding capacity.

If peers are completely altruistic, they can all receive the source rate. This is shown in Figure 1(a), where the Ethernet peer contributes 1600Kbps to the overlay. However, if peers contribute only as much as they receive (the *Bit-*

for-Bit semantic), then the Ethernet peer contributes just 400Kbps as shown in Figure 1(b). With limited forwarding capacity, only one DSL can receive the video stream.

We view the level of altruism as a continuous spectrum. The level of altruism determines the forwarding capacity in the overlay, and thus the receiver performance. For example, if the Ethernet peer contributes twice the source rate, the overlay can in turn accept two DSL peers, as shown in Figure 1(c). Thus the performance is better than (b) but worse than (a).

2.2 Multiple Description Codec

Content publishers typically want to accept as many viewers in the broadcast overlay as possible, and it is not desirable to reject peers as shown in Figure 1(c). In an overlay environment where the forwarding capacity is fixed, we can reduce the number of rejection by giving some peers lower video bitrate, but the resulting video quality is still acceptable to them. For example, we can accept all the DSL peers if the Ethernet peer forwards 200Kbps video stream to each DSL, as shown in Figure 1(d). However, this requires a scalable video codec, such as multiple descriptions coding (MDC), that allows peers to receive different video bitrate.

Several overlay multicast protocols have used MDC to support receiver heterogeneity and improve resilience to machine failure and network congestion [6, 2, 4]. We use MDC for a different purpose. MDC gives peers the flexibility to contribute and receive small increments of bandwidth, instead of discrete increments of the source rate. Such flexibility helps us to express the altruism policy in finer granularity. For example, if each stripe of video is 50Kbps, then the Ethernet peer can contribute 450Kbps, and one DSL peer would receive an additional 50Kbps.

2.3 Alternatives to Single Tree

The overlay structure constructed by the protocol affects the efficiency in utilizing the valuable forwarding capacity. In the previous examples (Figure 1 (a)-(d)), the overlay structure is a tree, and the forwarding capacity at the leaf nodes cannot be used. Various protocols [4, 6, 2] build alternate overlay structures, such as mesh or multiple disjoint trees, to improve bandwidth utilization. As an example, consider a mesh structure in Figure 1(e). The DSL hosts can receive additional 100Kbps by forwarding non-overlapping data to each other. In the next section, we assume the overlay runs an efficient protocol that can utilize *all* forwarding capacity (subject to MDC channel granularity). In Section 4, we show how to incorporate policy constraints into a protocol that constructs multiple disjoint trees structure.

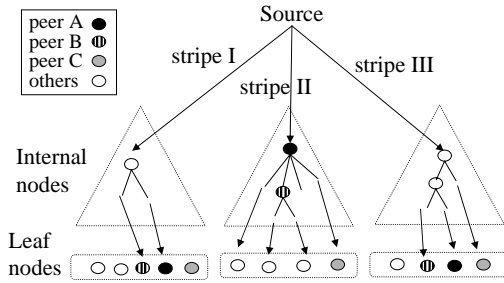


Figure 2: An example of a multiple disjoint tree structure with 6 receivers.

3. POLICY FRAMEWORK

In this section, we formulate a framework that can capture a range of policies with varying degrees of altruism. These policies define how much bandwidth peers should contribute and receive in return. We emphasize that this policy framework is simply a *reasonable* one used as a basis for our discussion, and we believe the policies should be customized to specific application requirements and peer environments when appropriate.

Assumptions: We assume the content publisher is responsible for setting the (minimum) level of altruism that peers must behave in the broadcast overlay. The publisher enforces the altruistic behavior by the software agent that runs the overlay protocol on users' machines. We assume users are *obedient*, and they do not change the protocol behavior by circumventing the software. However, obedient users can choose not to join the overlay if the level of altruism is higher than they can tolerate. Most popular peer-to-peer applications, such as Kazaa [5], make these assumptions.

Definition: Let N be the set of peers participating in the overlay. For peer $i \in N$, let F_i be the forwarding capacity and R_i be the receiving capacity. Both F_i and R_i represent the upper bound bandwidth that an overlay protocol can potentially utilize. These values are bounded by the access bandwidth of peer i and may be lower with competing Internet traffic. For simplicity we assume these values are statically configured. In a real implementation measurement techniques can be used to dynamically estimate the capacity values.

An altruism policy defines the actual amount of bandwidth peer i should contribute (f_i) and receive in return (r_i). We express the level of altruism as a single parameter K , where higher K means a higher level of altruism. We first show how to assign f_i and then r_i . Note that r_i receives either the *entitled bandwidth* (r_i^+) or the *social bandwidth* (r^*), whichever is higher.

Contributing Bandwidth (f_i): A peer should never contribute more than its forwarding capacity, and the contribution should be bounded by the maximum bandwidth it can receive (with a source rate of s_{max}), and the level of altruism.

$$f_i = \text{MIN}(\text{MIN}(s_{max}, R_i) * K, F_i) \quad (1)$$

Entitled Bandwidth (r_i^+): If a peer contributes more bandwidth, it should be "entitled" to receive more in return. Moreover, the ratio of the contributing and entitled bandwidth should depend on the level of altruism. This

	f	r ⁺	1 st stripe	2 nd stripe	3 rd stripe
Peer A	200	100	interior node (0)	entitled leaf (0)	social leaf (3)
Peer B	100	50	interior node (0)	social leaf (2)	social leaf (3)
Peer C	0	0	social leaf (1)	social leaf (2)	social leaf (3)

Figure 3: An example depicting bandwidth allocation of three peers ($K=2$ and each stripe is 50Kbps). The shaded blocks are entitled bandwidth, and the numbers in brackets are their priority in receiving n th stripes from the social pool.

ratio is shown as follows:

$$r_i^+ = f_i / K \quad (2)$$

Now we explain why K must be at least 1. Intuitively when $K < 1$, the "demand" of bandwidth exceeds the "supply" of bandwidth. That is, the total amount of entitled bandwidth exceeds their contributing bandwidth (discounting the source). When $K = 1$, the bandwidth supply equals the bandwidth demand. In this case, peers are completely selfish (Bit-for-Bit), and they contribute exactly as they are entitled to. When $K > 1$, there will be some bandwidth capacity left after all the entitled bandwidth are allocated. We call this the *social pool* of bandwidth. Next we describe how to assign this social pool.

Social Bandwidth (r^*): To optimize for performance, we continue allocating the social pool until either there is no more bandwidth left, or all peers have reached their maximum receiving capacity. There are many ways to assign the social pool to peers. In this paper, we choose a policy that uses the social pool to maximize the minimum of r_i for all peer i . We call this the *social bandwidth* (r^*). In another word, given a fixed peer environment, all peers receive either the social bandwidth r^* or the entitled bandwidth r_i^+ , whichever is greater.

$$r_i = \text{MAX}(r_i^+, r^*) \quad (3)$$

Derive r^* : r^* depends on the contributing bandwidth of all the peers in the overlay. If all information is known, we can compute r^* directly, as shown in Equation 4. Note that i^* is the number of peers who receive the social bandwidth of r^* . In a distributed protocol where peers dynamically join and leave the overlay, it is difficult estimate r^* . In Section 4, we give a distributed protocol where the social pool is iteratively allocated until r^* is reached.

$$\sum_{i=0}^N f_i = \sum_{i=1}^N r_i = \sum_{i=1}^{i^*} r^* + \sum_{i^*+1}^N r_i^+ \quad (4)$$

Example: Figure 1(e) depicts an example where $K = 2$ and the maximum received rate is 400Kbps. Each DSL peer contributes all of its forwarding capacity (100Kbps), and is entitled to receive 50Kbps. However, the Ethernet peer contributes just 800Kbps (even though its capacity is 1600+Kbps), and is entitled to receive 400Kbps (full source rate). After the entitled bandwidth is allocated, the social bandwidth pool is divided among the DSL peers. In this example, r^* is 300Kbps, and this is the actual bandwidth the DSL peers receive.

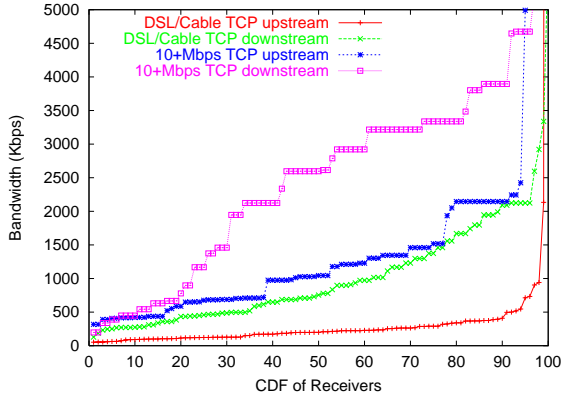


Figure 4: Measured TCP throughput of peers in Slashdot.

4. DISTRIBUTED PROTOCOL EXTENSION

This section presents one way to implement the altruism framework in a distributed protocol. We leverage ideas from protocols that build a multiple disjoint tree structure [2, 6]. In the original multiple tree proposal, the source splits the video stream into m equal stripes using MDC and multicasts each stripe along a separate tree. Each peer selects *one tree* at random, and joins the trees as an *interior node*. It joins all other trees as *leaf nodes*. Figure 2 illustrates an example structure that a multiple tree protocol constructs. Note that a peer contributes bandwidth to one of the m trees, and receives bandwidth from *all* the trees it joins, including the one it contributes bandwidth to.

Now we attempt to incorporate the taxation semantic into this protocol and motivate the need to extend the protocol. Let v be the bitrate of one stripe. To forward f_i bandwidth, peer i would configure the fanout of the interior node to be f_i/v . To receive r_i bandwidth, the peer would join r_i/v trees. The issue is that r_i depends on the social bandwidth (r^*), which requires global knowledge of all other peers. In the following, we describe a technique to infer r^* in a distributed fashion.

Conceptually, the distributed protocol incrementally allocates bandwidth to peers (by increasing r^*) until there is no bandwidth left in the social pool. First, the heuristic assumes r^* is 0, and allocates r_i^+ (entitled bandwidth) to the peers. After all peers receive their share of r^+ , the leftover bandwidth in the system is the social pool. Next, the protocol iteratively increases r^* by 1, until the social pool is exhausted. With every unit increment of r^* , peers who currently receive less than r^* can join one more tree.

To facilitate this join order, the peer assigns a *priority* value for each of the m trees it joins. The peer marks the first r^+ joins with the highest priority. Then the peer iteratively marks all other joins with decreasing priority. Table 3 shows an example of priority assignment for the three peers, where $K=2$ and each stripe is 50Kbps. To receive the entitled bandwidth of 100Kbps, peer A sets a high priority (priority 0) to join the first 2 trees, in which one is an interior node and the other is a leaf node. A has a lower priority (priority 3) to join the third tree until all other peers join two trees. In another word, A will not receive the third stripe until r^* becomes 150Kbps. In the case where two peers have the same priority and contend for one “spot” in

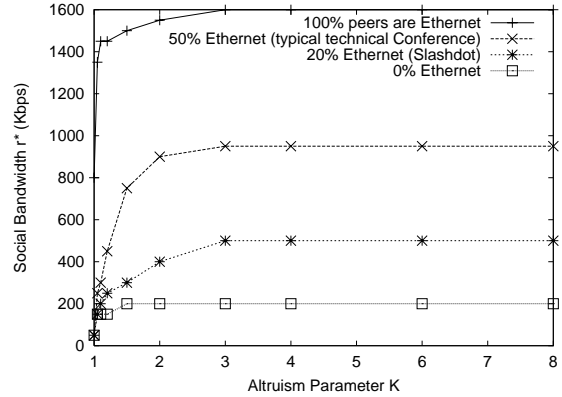


Figure 5: Impact of altruism on social bandwidth for various peer environments.

a tree, an arbitration rule is needed. In our implementation, we favor the peer with higher f . So A has higher priority to join the third tree before B does, which rewards peers who contribute more.

Now that the nodes in each tree have proper priority values that reflect the altruism parameter, the protocol needs to perform admission control based on the priority. Nodes that are accepted in the tree should have higher priority than those rejected. To achieve this, each interior node individually runs a *preemption* rule on the joining peers. If the fanout bound is reached, a peer with a higher priority can preempt existing peers with lower priority. Though we do not formally prove here, by induction a peer i is rejected from a given tree only if all other peers in the tree have equal or higher priority than i in the steady state. With dynamic peer environment, a peer that was previously preempted may become eligible. Thus a preempted peer periodically (every 30 seconds) attempt to rejoin the tree and get its fair share of r^* .

5. EVALUATION

In this section, we present preliminary results that support the following two assertions. First, the level of altruism is an important parameter in overlay streaming. We show that even a small level of altruism (K) can significantly improve the social bandwidth (r^*) in the overlay. Second, it is feasible to design an efficient overlay multicast protocol that implements a wide range of altruism policies. We evaluate the protocol presented in Section 4, and show the performance is close to an ideal (centralized) algorithm under a range of K values.

5.1 Simulation Setup

We consider two important factors that affect the protocol performance: (i) the heterogeneity of peers in their forwarding and receiving bandwidth capacity, and (ii) the join and leave dynamics of peers. The effect of altruism is more pronounced when peers are heterogeneous and their forwarding and receiving capacity is different. High group dynamics puts greater stress on the distributed protocol. To realistically model these two factors in simulation, we use traces from several live broadcast events using peer-to-peer overlays [3]. Due to space constraint, we show the results of one trace (Slashdot) in detail. The Slashdot event attracts

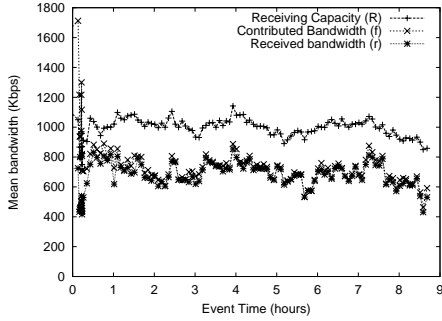


Figure 6: Mean bandwidth of peers vs. time in the Slashdot trace simulation. K is set to 2.

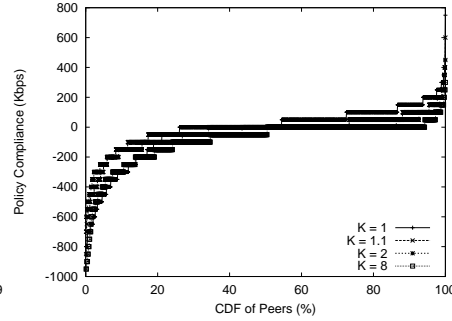


Figure 7: Cumulative distribution of policy compliance for all peers in the experiment.

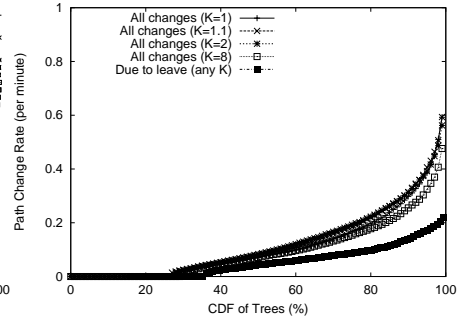


Figure 8: Cumulative distribution of path change rate for all the trees peers in the experiment.

1316 peers, the largest among all the traces. The mean and median join time is 18 minutes and 3 minutes, respectively. The trace lasts for 8 hours. The peak group size is 160.

Modeling Bandwidth Capacity: The trace also contains the bandwidth measurements that helps us model the forwarding and receiving capacity of peers. For each peer, it measures the TCP throughput to some well-provisioned servers and their access bandwidth. The access bandwidth measurements indicate that about 20% of the peers are behind 10+Mbps links and 80% are behind DSL/cable modem. For a finer grained and conservative estimation, we use the TCP throughput to model peers bandwidth capacity. Figure 4 shows a cumulative distribution of peer TCP upstream and downstream throughput. We observe significant heterogeneity among peers, and the upstream bandwidth is significantly lower than the upstream bandwidth. We use these values in our simulation unless otherwise noted.

Other Simulation Parameters: Our simulator assumes uniform delay between any pair of peers and no packet loss. The maximum streaming rate is 1600Kbps. The stream is evenly divided among 32 stripes using MDC, and each strip is 50Kbps.

5.2 Importance of Altruism

We first assume an ideal (centralized) protocol, and study the performance impact of altruism by varying K . We consider two categories of hosts, DSL and Ethernet, and repeat the experiments for various composition of these peers. For example, the Slashdot environment has approximately 20% peers behind Ethernet and 80% behind DSL. On the other hand, a typical technical conference environment has 50% peers behind Ethernet. We consider 1000 hosts, and assign them to appropriate category (DSL or Ethernet) to reflect the composition ratio. Their bandwidth is drawn from a distribution shown in Figure 4. The protocol then computes the receiving bandwidth directly for the given altruism parameter. The performance metric is social bandwidth (r^*), which represents the minimum bandwidth any peer should receive in the environment.

Figure 5 shows the r^* achievable for a given K . Each curve represents one peer composition. We make two observations. First, even a small level of altruism can significantly improve r^* for all the environments. In Slashdot, r^* rises quickly from 50Kbps to 400Kbps when K increases from 1 to 1.5. However, there is no visible benefit to make K larger

than 4. Second, the impact of altruism is more pronounced in a more heterogeneous environment. For example, moving K from 1 to 1.5 improves r^* to 750Kbps in a heterogeneous environment with 50% DSL, but the improvement is up to 200Kbps in a homogeneous environment with 100% DSL. This is because in a homogeneous environment, there is either fundamentally not enough forwarding capacity to contribute, or too much capacity that most peers can contribute full source rate.

5.3 Protocol Performance

Does our distributed protocol work well under a wide range of altruism parameter? We study this question by selecting 5 sample K values between 1 and 8, and conducting simulation based on the Slashdot trace. The simulator captures the overlay tree changes due to peers joining and leaving the group, but not due to network congestion. So peers would switch parents only if they are preempted by other peers, or if their parents leave the group. To find a parent, a peer probes a small number of other peers (up to 5) that are the interior nodes in the tree. This limit bounds the overhead in maintaining each tree. The protocol is evaluated with the following three performance metrics:

Utilization Ideally, an efficient protocol should utilize all the bandwidth peers contribute to the overlay. However, a distributed protocol cannot be as efficient because it takes some time for peers to (i) find unsaturated trees and the parent with unused fanout, or (ii) find parents who will preempt other children because the joining peers have higher priority. Figure 6 shows the mean bandwidth metrics of peers as a function of experiment time when $K=2$. The top curve (R) is higher than the other two curves (f and r) most of the time. This indicates that there is *not* sufficient contributing bandwidth to given all peers a full source rate. Our protocol is quite efficient, which utilizes 95% of the contributed bandwidth most of the time, as shown by the difference between the two lower curves. There is a visible dip in utilization at the beginning of the experiment. This is because the group size is small at the same compare to the number of trees in the overlay. As a result, the aggregate forwarding capacity can vary greatly among trees. Thus one tree can be saturated while others are not. We currently investigate ways to dynamically balance forwarding capacity among trees.

Policy Compliance A good protocol should not only utilize the contributed bandwidth efficiently, but also allocate

the right amount of bandwidth to peers in compliance with the policy. To measure compliance, at every time interval (5 seconds) we compare the difference between the received bandwidth allocated by our protocol and the bandwidth the peer should ideally receive. Figure 7 shows a cumulative distribution of that difference for all peers during the experiment. A positive difference means the protocol allocates more bandwidth than it is supposed to. Each curve represents one K value. We observe that a majority of time (70%), peers see a difference no more than 100Kbps, indicating the protocol is fair most of the time. This is true for all the K values evaluated. The curves show a tail as low as several hundred Kbps, indicating that peers at some point receive significant less than they should. This is mainly the artifact of the inefficiency in allocating bandwidth.

Stability Our protocol achieves high utilization and compliance through preemption. However, preemption comes with a cost in performance. Peers that are preempted may experience transient data loss before finding another parent. Worse yet, if interior nodes are preempted, their descendants are also affected. To capture the cost of preemption, we count the rate of *path changes* for each tree that peers participate during the experiment. Some of the path changes are fundamentally unavoidable. Specifically, the path to a peer will change if any one of their ancestors leaves the group.

Figure 8 shows the cumulative distribution of path change rate for each tree that peers participate in the experiment. Each curve represents one K value. There are two points to take away from this figure. (i) K has little effect on the path change rates. The rates are slightly lower with higher K , because the trees are wider with more fanout in the interior nodes. (ii) The total cost of implementing the policy is twice the fundamental cost in maintaining the structure. This is shown in the difference between the top curves and the lowest curve. The lowest curve indicates the path change rates due to peers leaving the group. The rates are very similar for all K and we show only one curve here. A majority of the path changes due to preemption is in peers “chasing” the unused social bandwidth pool, which can change with peers joining and leaving the group.

6. RELATED WORK

To our knowledge, this is the first work that explicitly considers altruism for peer-to-peer streaming, and proposes a protocol design that can implement a wide range of altruism parameter. Below we review related literature on overlay multicast protocols. These protocols fall into one of the two categories based on their assumption about the degree of altruism of peers.

ESM [3] and Bullet [4] assume peers are completely altruistic, meaning that the overlay protocols can use all the available bandwidth resource at end systems. Their objective is to deliver the highest multicast throughput to all end systems by harnessing the bandwidth effectively. The main difference between the two is in the overlay structure they construct. ESM constructs a tree rooted at the multicast source. Bullet constructs a (sparse) mesh, where data is disseminated along all mesh paths. By utilizing more links, a mesh structure has potential to gain better throughput and data resilience. Despite the difference in structuring the overlay, neither protocol provides an explicit mechanism to reward peers who contribute more bandwidth.

CoopNet [6] and SplitStream [2] assume peers are com-

pletely selfish. The protocols implements the Bit-for-Bit policy, where peers receive only as much bandwidth as they contribute. The key difference between the two protocols is that CoopNet uses a central server to coordinate the tree construction while SplitStream builds the trees distributively. Hosts can contribute more than their receiving bandwidth in SplitStream. However, the extra bandwidth is mainly used to ensure the success of tree construction during transient node failure. Our results show that Bit-for-Bit policy has suboptimal performance in a heterogeneous environment, and even a small degree of altruism goes a long way in improving hosts behinds asymmetrical links with low upstream bandwidth.

7. SUMMARY

In this paper, we argue that it is important to explicitly consider altruism in a peer-to-peer streaming protocol. We express altruism as a parameter K , which is the ratio of the bandwidth a peer contributes and receives in return. We demonstrate that a small level of altruism ($K=1.5$) can significantly improves the overall performance of peers. Moreover, the benefit is more pronounced in a more heterogeneous environment. To construct overlay to achieve a target altruism level, we extend the protocols that construct multiple disjoint trees. Such a structure enables us to express policies flexibly and in fine granularity. The trace simulation based on Slashdot shows that, for a wide range of altruism parameter, the protocol can achieve high utilization while being compliant to the target altruism level. The protocol consistently utilize above 95% of the bandwidth and the allocation is compliant to the policy within 100Kbps 80% of the time. However, this comes at a cost of structure instability. We show the cost is twice the fundamental cost in maintaining the structure.

While we show the importance of altruism in peer-to-peer live streaming, it is not clear whether the same concept applies to other peer-to-peer applications such as file sharing. Peer-to-peer streaming seems unique in two ways. (i) The application “naturally” brings out altruism in people. Live broadcast is typically associated with a specific event people care about. Though altruism is also observed in file sharing, we believe people will likely be more altruistic toward an event (e.g. broadcast of Chinese New Year celebration) than a system (e.g. KaZaa). (ii) Live streaming has more stringent performance requirement. A file is still usable if it takes three times as long to download, but a video stream may not be watchable if the bitrate is three times less.

8. REFERENCES

- [1] E. Adar and B. A. Huberman. Free-riding on Gnutella, 2001. *First Monday* 5(10).
- [2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of SOSP*, 2003.
- [3] Y. Chu, A. Aganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early Experience with an Internet Broadcast System. In *USENIX Annual Technical Conference*, 2004.
- [4] J. A. D. Kotic, A. Rodriguez and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of SOSP*, 2003.
- [5] Kazaa. <http://www.kazaa.com/>.
- [6] V. Padmanabhan, H. Wang, and P. Chou. Resilient Peer-to-peer Streaming. In *Proceedings of IEEE ICNP*, Nov. 2003.