The Importance of History in a Media Delivery System

Richard J. Dunn, Steven D. Gribble, Henry M. Levy, John Zahorjan
University of Washington
E-mail: {rdunn,gribble,levy,zahorjan}@cs.washington.edu

Abstract

We examine the performance dynamics of a generic media delivery service, in which a single provider distributes to a set of peers using a BitTorrent-like protocol. Previous studies of BitTorrent have examined its performance in delivering single objects; here we discuss the dynamics of distributing a population of objects. Specifically, we assume that peers cache objects they download, and examine strategies for choosing which objects peers should serve while participating in the system. We examine a set of strategies, and show that a simple strategy yields performance gains over that employed by current BitTorrent-like systems.

1. Introduction

In recent years, the BitTorrent protocol has proved an efficient means to distribute content from a single provider to a set of interested peers. Its success is such that recently several proposals [5, 9, 8] have been made to utilize this protocol as part of a media delivery service, delivering non-real-time media content (movies, music). Such a service utilizes the interested peers to reduce bandwidth demands on the content provider.

In this paper, we examine the performance dynamics of such a system. We are interested in how to manage the limited resource of the peers in the most efficient manner. Previous studies of BitTorrent have focused on its performance in delivering a single object. Here we focus on how to manage resources across a wide population of objects. Specifically, we assume that peers cache downloaded content for some time, and seek to determine which objects peers should serve while participating in the system.

We approach the problem by simulating a generic media delivery system. Using a model developed from traces of P2P file-sharing systems, we drive a simulation with a request pattern that exhibits the characteristics of measured requests to media objects. Using this simulation, we determine performance based on the bandwidth requirement placed on the content provider.

Our key result is to show that good performance in media delivery relies on clients caching and re-distributing objects they download, referred to as "seeding" in BitTorrent parlance. We show the the current default behavior of many BitTorrent clients (seeding the last object downloaded), is far less than optimal, and that a simple change in seeding strategy can yield tremendous performance improvement.

In the next section, we examine the structure of our simulation, outlining our models of requests, peer availability, and object delivery. In Section 3, we examine the best and worst possible performance of any strategy, and the performance of a strategy implicitly used by existing BitTorrent clients. In Sections 4 and 5, we examine improvements on this strategy, and conclude in Section 6.

2. Methodology

Our approach is to simulate a generic media delivery system and to measure the effect that certain design choices have on its overall performance. We begin by framing the general structure of a media delivery system:

- There is a single content provider, who possesses a set of objects. The provider has unlimited bandwidth with which to serve these objects (more precisely, the provider has enough bandwidth to serve to all peers at their maximum download rate, consistent with a well-provisioned service.)
- There is a set of peers, who over time make requests on and download objects from the provider. Peers have fixed download and upload bandwidths, and we further assume that download bandwidth exceeds upload bandwidth by some constant factor.
- The provider faces a fundamental trade-off between the bandwidth it expends to deliver the objects to the peers, and the rate (and therefore latency) at which peers can download an object. To resolve this trade-off, we assume the provider serves objects such that each peer can download at its full download bandwidth, regardless of the total bandwidth expended by the provider. This is consistent with providing a good user experience; our goal is to reduce bandwidth through other means.
- We assume that the peers are willing to use their upload bandwidth to relieve the bandwidth requirements on the provider. Peers can serve objects they are currently downloading or have downloaded in the past, as long as they are online in the system. We assume peers are online whenever they are downloading an object and at select other times, described below.

We simulate the dynamics of this system, with peers coming on- and offline, making requests, and retrieving objects. We drive the simulation with a time series of requests from the peers. Aside from the general structure of the system described above, we require three key inputs:

1. A model of requests. We use this to develop the time-series pattern of requests from the clients to the object population. For this, we will utilize an artificial stream generated by a previously published model based on traces of existing P2P systems.

2. A model of peer availability. This governs when each peer is offline, online and downloading an object, or online without actively downloading. We rely on a simplified model based on standard usage of BitTorrent systems.

3. A model of efficient delivery of objects. This governs how we use the resources of the content provider and the peers to distribute individual objects. We assume an idealized version of BitTorrent based on published studies of its performance.

In the next three sections, we discuss the details of each of these models separately.

2.1 Model of Object Requests

We utilize the model first proposed in [3], and extended by [2]. This model was developed to mimic the properties of request streams seen in measurements of various P2P file-sharing systems. The model takes as input a per-peer request rate, peer and object population size, and a rate of introduction of new objects, and produces request streams of arbitrary duration. The generated request streams exhibit patterns of requests with appropriate distribution of those requests across objects, across peers and, for each object, across time.

Several properties of these distributions are key to our results. First, requests are distributed across objects following a power-law (or Zipf-like) distribution, in which a few objects receive a large portion of the requests, but unpopular objects as a whole also receive a large portion of requests. Requests across time to particular objects exhibit a peak of popularity when an object is introduced into the system, waning as the object ages. Also, new objects tend to start as popular objects, displacing existing popular objects.

For the simulations that we show results for in the next few sections, we assume a peer population of 100,000, an initial object population of 70,000, a request rate of 40,000 per day (and thus each peer makes a request every 2-3 days), and an object arrival rate of 10 per day. Objects are all around the standard size of movies (700 MB). We simulate using a 100-day long request stream. (Though our quantitative results reflect these chosen numbers, reasonable variations from these values do not qualitatively change our conclusions.)

2.2 Model of Peer Availability

We adopt a simplified model of peer availability based on the current perceived usage of BitTorrent. Namely, a client will become available when it wishes to make a request to the system (governed by the request model). The client will remain available as it downloads the object. After it has obtained the full copy of the object, it will remain available for some period of time, which we call the "seed time" and which is the single parameter for the availability model.

Two studies of BitTorrent systems [6, 4] note that the vast majority of clients follow the pattern described above, with seed times on the order of hours. This model also agrees with anecdotal evidence of how users interact with BitTorrent objects. Since the download is not a real-time stream, users will often "fire and forget" allowing the download to proceed and checking back later that it has finished. Torrent sites also often encourage users to seed after their download has finished. In our experiments, we vary the average seed time between 3 and 15 hours after each download.

2.3 Model of Object Delivery

We assume an idealized BitTorrent model of object delivery, with additional unicast delivery of objects to fill each peer's download capacity. Performance analyses of BitTorrent [7, 1] have shown that it achieves a near-optimal efficient use of peer upload capacities. For the purpose of a single provider distributing an object, this means that up to a particular upload rate (equal to the average upload bandwidth of the clients), each peer (regardless of the number of peers) will receive that download rate, making delivery to more than one peer essentially "free"

In a world with asymmetric bandwidths and content providers with excess bandwidth, however, we may do better by having the content provider use less-efficient unicast delivery to fully fill the excess download capacities of users. Based on our assumptions earlier, we assume our content provider provides distributed delivery up to the peer upload rate, and unicast delivery beyond that until all peers are downloading at capacity.

To achieve the above effect, all peers downloading an object must also devote their upload bandwidth to that object (this condition is implicit in the analyses cited above). Therefore, we assume that while a peer is downloading, it is seeding that object, but that during the seed time (no active download), it is free to seed any object it has previously downloaded. If we assume N peers are downloading an object, M peers (above the N downloading) are seeding it, and each peer has download capacity D and upload capacity U, the demand on the content provider for that object is then:

$$\mathtt{BW} = \mathtt{U} + ((\mathtt{D} - \mathtt{U}) * \mathtt{N}) - (\mathtt{M} * \mathtt{U}) \tag{1}$$

To reflect real-world asymmetries in download versus upload capacity of peers, we set $D/U = 3$, specifically $D = 60$KB/s, $U = 20$KB/s. (As above, changing these values or their ratio does not quantitatively affect our conclusions.)

2.4 Bringing it Together

Given the models above for requests, availability, and delivery method, we can outline the design of the system simulation as a whole. We begin with a population of P peers, all initially offline. As time progresses, we receive requests from peers. If a peer p makes a request at time t, it then comes online and begins downloading the object at its download capacity D. When the object is fully downloaded, the peer begins seeding some object that it has previously downloaded, and will continue to do so for the seed time S. The peer will then go offline until it makes its next request.

Using the notation of Equation 1, the total demand on the content provider (across all objects i) is $B_{provider} = \sum_i U + ((D - U) * n_i) - (m_i * U)$. Our goal is to minimize $B_{provider}$. We assume that most of the parameters in our model are fixed. That is, we have no control over the request pattern, object population, delivery model, or download and upload bandwidths of peers (and we believe this to be a reasonable real-world assumption). Our remaining control over the system is limited to the seeding strategy of peers (i.e. which object each peer should seed) and (to a likely far lesser extent) the average seed time for peers. Thus our goal in this paper is to explore the space of seeding strategies, evaluating which ones most effectively minimize the bandwidth requirement on the content provider.

2.5 Experimental Approach

Over the next few sections, we will evaluate the performance (demand on the content provider) of several different seeding strategies. We begin by first bracketing the achievable performance of any seeding strategy. We show the worst-case performance, namely the demand on the content provider when peers contribute no seeding resources. We then show a measure of the best possible performance, given the constraints of peer upload capacities and histories of downloaded objects.

With a minimum and maximum in place, we show the performance of the existing BitTorrent strategy, in which peers seed the object they most recently downloaded. To improve on the performance of this strategy, we then consider two types of approaches. The first (Section 4) are object-centered approaches, in which we attempt to move peers around to optimize the performance of a single object at a time. The second (Section 5) are peer-centered approaches, in which each peer independently decides which object to seed. Our results compare each proposed strategy to the existing strategy and the minimum and maximum possible performance.

3. Basic Performance of the System

We first consider the basic dynamics of the system, and then examine the performance using a seeding strategy based on common usage of BitTorrent. As stated in the previous section, our measure of performance for our delivery system is the total bandwidth demand on the content provider, after accounting for bandwidth provided by the peers. We first evaluate the minimum and maximum bandwidth demands we expect to see at the content provider: these provide boundaries for any seeding strategy we evaluate. We then evaluate the performance of a seeding strategy employed by most BitTorrent client applications.

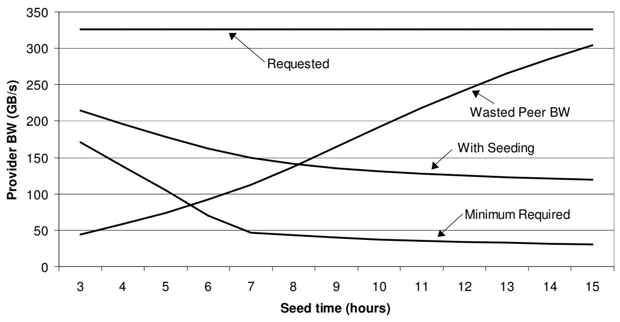

Figure 1. Bandwidth vs. seed time (last-object seeding). *This shows the demand on the central content provider across a variety of peer seed times. 1) Requested: total demand, 2) Minimum required: minimum demand possible, 3) With seeding: using a last-downloaded seeding strategy, 4) Wasted Peer BW: unused peer capacity using last-downloaded strategy.*

3.1 Bounds on System Performance

There is clearly a maximum on this requested bandwidth; at any time, some finite number of peers P_d are actively downloading objects. Since we assume all peers download at rate D, the maximum demand on the content provider is $P_d * D$. The line labeled "Requested" in Figure 1 show this maximum requested bandwidth at a single point in time (day 50) of our generated trace, across a variety of seed times ranging from 3 to 15 hours. We see that the value is constant; this is expected, since we are considering maximum requested bandwidth, which is unaffected by any bandwidth provided by the peers.

The minimum bandwidth demand is harder to account for. Recall that our bandwidth demand is lowered by peers in their seeding phase providing bandwidth to the downloading peers. A logical conclusion is that if more peers are in their seeding phase, more bandwidth will be offloaded from the content provider, driving its bandwidth towards zero. Two factors play a role in the effect that seeding peers have:

1. We can only reduce the bandwidth on the content provider to the extent that there are seeding peers in the system. That is, there is a maximum upload capacity that the set of seeding peers can provide, and we can only increase performance by this value.

2. However, even if sufficient absolute bandwidth exists on the seeding peers, the peers cannot serve objects they do not have. For any particular object, it is possible that the number of peers who have previously downloaded the object and are now in their seeding phase is insufficient to satisfy the demand on that object. Across all the objects, the sum of these deficits correlates the maximum performance improvement.

As stated, both these factors independently determine a minimum demand on the content provider. The maximum of these two factors is thus an absolute minimum of the demand.

The line labeled "Minimum required" in Figure 1 shows this minimum value, again across a variety of hold times. times. We see that the minimum is far below the maximum requested bandwidth and, as expected, improves with seed time. Note the sharp knee at a seed time of 7 hours. This represents the point at which the two minimums trade dominance. At fewer than 7 hours, the first minimum dominates, since these low seed times introduce very little total peer bandwidth. However, the situation improves very quickly, and after 7 hours the second minimum dominates.

3.2 Performance of a Basic Seeding Strategy

Given a maximum and minimum demand on the content provider, we can now evaluate the performance of specific seeding strategies. In this section, we consider the current model of BitTorrent seeding, namely, to seed the last object downloaded. While this strategy is not inherent in the design of the BitTorrent protocol, most implementations of BitTorrent clients foster this behavior:

- Several BitTorrent client applications (ex. the canonical BitTorrent client, BitComet) use the concept of single application instance per torrent file (and object). Once this instance is closed for some reason, there is no incentive to re-open that particular file, and thus older objects are unlikely to be seeded.
- Even in the case of one application instance handling several files at once, several applications (ex. μTorrent, Azureus) explicitly encourage the seeding of recently downloaded files over older files.

In our simulation of this strategy, a peer comes up at the time of a request, fully downloads the object, then seeds this object for the duration of the seed time.

The line labeled "With Seeding" in Figure 1 shows the demand on the content provider when the peers in our simulation adopt the last-downloaded seeding strategy. As we can see, utilizing client upload bandwidth can significantly reduce the demand on the central provider; however, there is also a significant gap between the demand seen and the minimum possible demand, suggesting alternate strategies might yield improved performance.

Additional analysis reveals that many peers have wasted bandwidth: that is, many objects have more seeders than is necessary to accommodate the demand on those objects. The wasted bandwidth is high enough that it should be possible to reduce the demand on the content provider to the minimum possible demand. In the next two sections, we introduce a variety of seeding strategies that attempt to do just that.

4. Object-centered Approaches

As mentioned at the end of the previous section, the key drawback to a last-downloaded approach is that certain objects (currently or very recently popular objects) have a surplus of peers seeding, at the expense of other (usually older) objects. To resolve this problem, we require a more intelligent scheme for choosing which objects a particular peer should seed. In this section, we consider object-centered approaches: that is, we will attempt to optimize the performance of particular objects by moving peers with wasted bandwidth to those objects.

Object-centered approaches are appealing because BitTorrent is an object-based system. The tracker represents a centralized point of information with regard to each individual object. Therefore, it is reasonable to assume that choices could be made there for optimization of an object. We consider two approaches, which use successively more information to achieve better performance.

4.1 Poaching

Let us define the "seeding balance" of an object. We take the seeding balance as the difference between the available peer seeding bandwidth and the demand on the object. If for a particular object there are M peers seeding, each with upload capacity U, and N clients downloading with capacity D, then the seeding balance is $M * U - N * D$). An object with a positive seeding balance thus wastes bandwidth (too many peers are seeding), while an object with a negative balance forces the provider to expend bandwidth. We will define a neutral balance as balance which is either zero, or positive with a value less than U (i.e. removing a single seeding peer from a neutral object will result in that object having a negative balance).

We first consider a method of seed selection we call poaching. In this method, an object with a negative seeding balance can "poach" a peer to act as a seeder for that object. Specifically, whenever an object undergoes a change in its balance (e.g. a new request, or a download finishes), the tracker will iterate over all peers that hold a copy of that object. For each of these peers, it will determine if they are currently seeding an object and, if so, whether that object has a positive seeding balance (and thus wasted seed bandwidth).

If so, the peer is tasked to seed the object in question, rather than its current object, simultaneously reducing the wasted bandwidth and the demand on the central seeder. The process is repeated until the object in question obtains a neutral balance, or no peers fit the criteria above. We modified our simulator to implement this strategy (i.e. optimize each object via the above method after all events affecting that object).

Figure 2 shows the bandwidth demand on the content provider when utilizing this new strategy. For comparison, we show the bandwidth requirements using the last-downloaded seeding strategy, and the minimum requirement, both from Figure 1. The line labeled "Poaching" represents the bandwidth demand under the poaching strategy described above. We see that poaching sharply decreases the demand, to 50–60% of that using the last-downloaded seeding strategy. Further, the total demand is very close to the minimum possible demand.

Implementing poaching would require that each tracker keep a history of all peers who have downloaded the object in the past, as well as a method for the tracker to

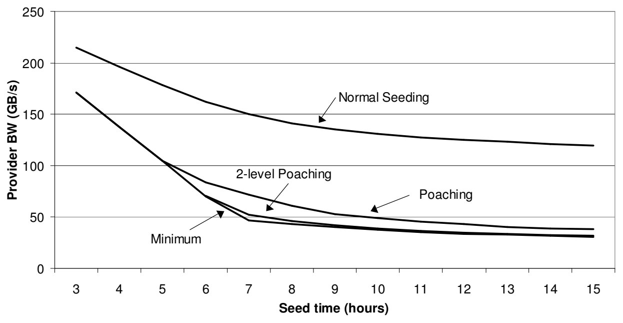

Figure 2. Performance of object-centered approaches. *Demand on the central content provider under a variety of object-centered approaches. 1) Minimum: minimum demand possible, 2) Normal seeding: using a last-downloaded approach, 3) Poaching: moving peers serving objects with positive seeding balances, 4) 2-level poaching: as above, using chains of two peers.*

query each peer as to its current seeding status. Maintaining the list of previous peers should be trivial; trackers already know who is currently downloading, and they need simply to not forget those peers when they finish. However, it may be infeasible for the tracker to contact peers which, unlike the trackers, may not have public IP addresses.

4.2 2-level Poaching

While the decrease in demand using a poaching strategy is significant, there is still some room for improvement, and it is useful to know why the above approach is not ideal. This is best illustrated by a simple example. Suppose we have three objects (O_1, O_2, O_3), and two peers (P_1 and P_2). P_1 hold objects O_1 and O_2, and is currently seeding object O_2. P_2 holds objects O_2 and O_3, and is currently seeding O_3. Finally, suppose object O_1 has a negative seeding balance, object O_2 has a neutral seeding balance, and object O_3 has a positive balance and thus wasted bandwidth.

Now suppose we wish to optimize object O_1. We cannot poach P_1, since the object it is seeding does not have spare capacity. And we cannot poach P_2 because it does not hold O_1. However, we could make a double change, and let P_2 seed O_2, which now gives O_2 enough spare capacity that we can poach P_1 to seed O_1.

We modified our simulator to take these possibilities into account by increasing the number of peers contacted when optimizing an object, and checking for the above condition. The line labeled “2-level poaching” (we consider switching the objects of two different peers at a time) in Figure 2 shows the bandwidth demand when we take this into account. We see that this approach essentially reaches the minimum possible bandwidth value; the remaining difference is likely due to extensions of the above scenario (i.e. changing 3 or more peers to achieve improvement.)

5. Peer-centered Approaches

In an object-centered approach, we focused on individual object, and made decisions about whether to re-task the potential seeds of that object to increase performance. In a peer-centered approach, we instead consider individual peers, and the decision becomes which object that peer should seed. Thus each peer independently determines which object to seed, regardless of the other peers' decisions.

Recall that an implementation issue for the object-centered approaches was that each tracker would need to maintain a history of peers that had downloaded the object, rather than just the current set of peers seeding or downloading the object. Further, the approaches required communication between the tracker and (potentially a great many) other peers. In a peer-centered approach, we need only maintain the list of objects a peer has downloaded (which we presumably would keep anyway). Further, there is no need for communication beyond querying objects for their seeding balance, and this operation is integral to existing BitTorrent systems (and trackers are by their nature publicly available). Therefore, if peer-centered approaches can provide similar performance gains as object-centered approaches, we will tend to prefer the peer-centered approach.

5.1 Best-object Seeding

We first consider a simple change to the current BitTorrent seeding strategy. Currently when a peer finishes downloading an object, it starts seeding that object immediately. We have shown that this results in wasted bandwidth, because very popular objects will have many more seeders than necessary. Thus it is likely that a peer will start seeding an object with a positive seeding balance, wasting bandwidth. So a better approach is that when a peer finishes downloading an object (and thus enters its seeding phase), it instead iterates over all objects it has downloaded in the past, and seeds one of the objects it holds that has a negative seeding ratio (if any). We call this approach best-object seeding.

Figure 3, like Figure 2, shows the minimum central provider demand and the demand of the current BitTorrent policy. The line labeled “Best-object” shows the effect of the change above. We can see that this approach shows some reduction in bandwidth, mostly at lower seed times. However, there is significant room for improvement when compared to the minimum.

5.2 Continuous Best-Object Seeding

The problem with best-object seeding as described above is that it still retains one of the problems of the current (last-downloaded) strategy, namely that we are committing to seeding a single object for a certain amount of time. It is quite possible that during that time period, the demand on the object will decrease to the point that our seeding is simply wasted bandwidth. Poaching in some sense overcomes this; if demand shifts such that a particular peer is wasting bandwidth, that peer becomes eligible to be poached.

Best-object seeding, while an improvement over last-

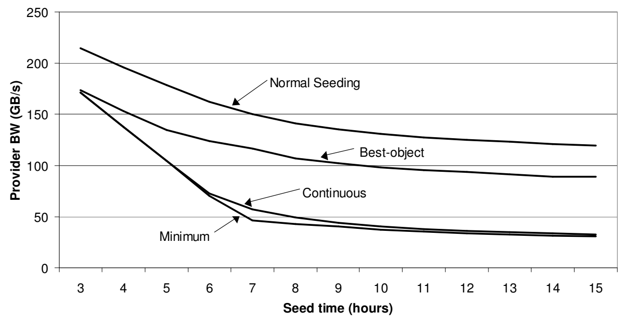

Figure 3. Performance of peer-centered approaches. *Demand on the central content provider under a variety of peer-centered approaches. 1) Minimum: minimum demand possible, 2) Normal seeding: using a last-downloaded approach, 3) Best-object: seeding the object with the worst seeding balance, 4) Continuous: Continuous update of object with worst seeding balance.*

downloaded seeding, does not have this property of poaching. However, a simple change is to reconsider the seeding choice at a finer granularity than a single seeding period. If we instead reconsider this decision every, say, 10 minutes, each peer would notice that it was wasting bandwidth, and shift to seeding a better object. We call this approach continuous best-object seeding.

The line labeled as such in Figure 3 shows the bandwidth demand when peers adopt this approach. We see that like 2-level poaching, this approach yields almost all the benefit possible: the bandwidth demanded is very close to the minimum required. This result, combined with the simplicity of this approach, implies that this technique is an obvious best choice for the seeding strategy of any BitTorrent client.

6. Conclusions

In this paper, we considered a specific P2P-based system design: assuming a central source of objects, we wish to minimize the demand on that source by utilizing the upload capacities of peers downloading those objects. Each peer uploads the object it is downloading until complete, and then donates a period of time in which is seeds some object it has previously downloaded.

We showed that the approach taken by most BitTorrent client of seeding the last downloaded object yields a significant reduction in demand on the source, but with significant room for improvement. Two approaches to optimization are to focus on objects (attempting to move peers to improve the performance of a particular objects) and on peers (each peer decides which object to seed based on its performance). We showed that the best of the object-centered approaches could yield near-ideal performance improvements, but only a great cost in terms of communication.

The best approach is peer-based, where each peer seeds any object in need of seeders, re-evaluating its choice regularly to minimize wasted bandwidth. This approach also yields near-ideal performance, but with only small changes necessary in the BitTorrent client applications, and little communication overhead.

A full evaluation of these methods is desirable. We simulated performance of the system by idealizing several portions, especially the BitTorrent-like delivery system. An implementation of such a system, and an emulation (or measurement of real-world usage) would serve to verify these results, though we are confident that our general conclusions will hold.

The main lesson of these experiments is that history matters: since popularity of media objects peaks early in their lifetime and gradually trails off, if we focus only on the currently popular objects (which last-downloaded seeding does) we end up wasting bandwidth with too many peers seeding an object which is waning in popularity. This comes at the expense of other objects, which have too few downloads to generate seeds using a last-downloaded strategy, but enough downloads that having seeds matters.

Though we have proposed a specific approach that works, in general we argue that a media delivery system either enforce or encourage clients to retain and seed objects they have downloaded in the past. This should not be difficult; it seems natural that clients will cache recently downloaded objects. The potential performance gains from not forgetting past objects are tremendous.

7. References

[1] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving a BitTorrent network's performance mechanisms. In *Proceedings of IEEE Infocom 2006*, Barcelona, Spain.

[2] R. J. Dunn. *Improving P2P Distribution of a Media Workload*. PhD thesis, University of Washington, December 2006.

[3] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, Lake George, NY, USA, October 2003.

[4] M. Izal, G. Urvoy-Kellera, E. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Proceedings of the 2004 Passive and Active Measurement Workshop (PAM)*, Antibes Juan-les-Pins, France, April 2004.

[5] S. McBride. Warner Bros. to try file sharing of films, TV shows in Germany. *Wall Street Journal*. January 30, 2006.

[6] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P file-sharing system: Measurement and analysis. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS 2005)*, Ithaca, NY, USA, February 2005.

[7] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM*, Portland, OR, USA, August/September 2004.

[8] G. Sandoval. BitTorrent inks licensing deal with studios. *CNET News.com*. July 10, 2006.

[9] G. Sandoval. BitTorrent inks studio distribution deal. *CNET News.com*. May 8, 2006.