BiToS: Enhancing BitTorrent for Supporting Streaming Applications

Aggelos Vlavianos, Marios Iliofotou and Michalis Faloutsos
Department of Computer Science and Engineering
University of California Riverside
Riverside, CA
Email:{aggelos, marios, michalis}@cs.ucr.edu

***Abstract*— BitTorrent (BT) in the last years has been one of the most effective mechanisms for P2P content distribution. Although BT was created for distribution of time insensitive content, in this work we try to identify what are the minimal changes needed in the BT's mechanisms in order to support streaming. The importance of this capability is that the peer will now have the ability to start enjoying the video before the complete download of the video file. This ability is particularly important in highly polluted environments, since the peer can evaluate the quality of the video content early and thus preserve its valuable resources.**

In a nutshell, our approach gives higher download priority to pieces that are close to be reproduced by the player. This comes in contrast to the original BT protocol, where pieces are downloaded in an out-of-order manner based solely on their rareness. In particular, our approach tries to strike the balance between downloading pieces in: (a) playing order, enabling smooth playback, and (b) the rarest first order, enabling the use of parallel downloading of pieces. In this work, we introduce three different Piece Selection mechanisms and we evaluate them through simulations based on how well they deliver streaming services to the peers.

I. INTRODUCTION

BitTorrent (BT) is a second generation peer-to-peer (P2P) file sharing protocol, designed by Bram Cohen [1]. During the last years, BitTorrent has been proved to be a very effective mechanism for P2P content distribution [2]. The success of BT lies on its ability to distribute content quickly by utilizing the capacity of **all** the peers in the P2P BT network. This ability comes from mechanisms that provide incentives to peers to contribute to the BT community preventing them from becoming Free-Riders. BT has attracted the interest of the research community, which through simulations or measurements try to capture the true characteristics of BT. Other studies, try to point out the effectiveness or the deficiency of the mechanisms of BT and provide ways to improve it. This intense interest strengthens our statement that currently *"BT is the king of the P2P protocols"*.

While BT has proved that it can successfully support distribution of time insensitive content, no work what so ever has been done in order BT to support time sensitive content. In this work, we will try to push the capabilities of BT to its limits, by studying if it can support content delivery with time constrains. Even though the current implementation of BT doesn't support this functionality, for reasons that are explained later, we try to add this functionality to the protocol with minimum modifications.

Can BT be modified to support streaming? This is the question we address here. BT was not designed for streaming media, and earlier works argue that BT in its current form is not suitable for streaming [3]. In our work we want to enhance BT with a **view-as-you-download** service. We want a peer to start reproducing the video content that is currently downloading, before it downloads the whole video file. This is very beneficial for the peer, because:

a) It reduces the time needed to start enjoying the video

b) It allows the peer to evaluate the quality of the video file without having to spend all its resources to download it first. This is particularly helpful in highly polluted environments [4].

In this work we propose **BitTo**rrent **S**treaming (**BiToS**), a protocol with the ability to support streaming based on BT. We identify the piece selection mechanism as the only thing that needs to be changed from the original BT protocol. BiToS becomes aware of the streaming order of the piece, thus preferring pieces that will be played soon. In more detail, it tries to strike the balance between downloading pieces in: **a)** playing order, enabling smooth playback, and **b)** the rarest first order, enabling the use of parallel downloading of pieces. An advantage of our approach is that it is tunable in that we can control the operation point between pure streaming and the original BT. Our preliminary simulations show that our approach is promising in providing streaming data in a timely fashion. In addition, we show that there exists a selection of system parameters that can give very good results.

The rest of this paper is organized as follows: In Section II, we describe briefly how the BitTorrent protocol works and its core mechanisms. In Section III, we present the related work. In Section IV, we identify what streaming services can be supported by the protocol with minimal changes, as well as we decompose BT and specify the exact mechanisms that need modifications. In Section V, we introduce our innovative BiToS protocol, while in Section VI, we present the experimental evaluation of our model. Finally, the paper is concluded in Section VII along with the future work.

II. BITTORRENT

BitTorrent's goal is to distribute fast and efficient large files by using the upload bandwidth of the downloading peers. BT is using swarming techniques, in which the torrent file (the content that is distributed), is split in pieces (typically 256KB in size).

In that way, peers can simultaneously download pieces from other peers. While the peer is downloading pieces of the file, it uploads the pieces that it has already acquired to its peers. Each time the peer has a new piece, it advertises this information to its peer set (the peers that the peer is connect to). The only centralized component of BT is an entity called *tracker*. The tracker is responsible to help the peers find each other and to keep the download/upload statistics of each peer. Moreover peers during their initialization they retrieve from the tracker information about the file, such as the number of pieces that the file is split, the hashes of each piece (for integrity verification), etc.

The strength of BT lies in its ability to resist to the Free-Riders phenomenon, in which selfish peers choose only to download the file without uploading. BT uses a *Tit-for-Tat* policy, where each peer chooses to upload to its peer as long as it takes something in return. If the neighbor peer behaves selfishly the *Choking mechanism* is invoked and the peer stops uploading to its neighboring peer. BT distinguishes peers into two categories, the *seeds* and the *leechers*. Seeds are peers that have already the whole file and leechers are peers that are in the progress of downloading the file. As soon as a leecher has downloaded the whole file, it becomes a seed.

Another vital mechanism of BT is the *Piece Selection mechanism*. Peers always select to download the rarest pieces within their peer set. This provides fast replication of the rarest pieces and ensures that the torrent file won't become easily extinct, in case a peer that has these particular pieces leaves. More information about the BitTorrent protocol can be found in [5].

III. RELATED WORK

In recent years, due to its popularity, BT has been the center of significant research. BitTorrent has been studied from different perspectives and many aspects of the BT application have been revealed. In [6] [7], they examine how well the incentive mechanism works and propose new simple mechanisms that can boost cooperation between peers even more. In [8] [9], the authors used the log file from a tracker in order to understand better the behavior of the BT peers, as well as the efficiency of the protocol in presence of flash crowds. From different perspective the authors in [10] instrumented a BT client and by gathering statistics and messages between their client and its peers tried to provide an analytical understanding of BitTorrent. Their findings show that BT is robust, efficient, realistic and inexpensive solution to the classical server based content distribution scheme.

A recent attempt [3] tries to provide streaming service by using a hybrid server/P2P streaming system approach. The clients retrieve the stream from a dedicated streaming server while in parallel share pieces using BT. The BT protocol remains almost unaltered with the only modification that clients won't download any data prior to the current playback time. This work differs from ours, due to the existence of the dedicated streaming server. In our approach we consider only BT as the primary mechanism for streaming. Moreover, in [3] they state that BT is not suitable for streaming. The stated reason is that peers will have only sequential pieces of the stream and thus Tit-for-Tat will fail. However in our work we show that by requesting pieces in a rarest first manner within a small window of the file, we can guarantee diversity of pieces as well as high QoS.

Another interesting work is CoolStreaming [11]. CoolStreaming, uses a data-centric design of an overlay network. Similar to our work, they introduce the notion of a sliding time window from which peers select to download a piece. The decision of requesting a particular piece is made based on a heuristic scheduling algorithm, which is similar to the Piece Selection mechanism found in BT. However, in contrast to our work, they use a fixed size sliding window without considering the adjustment of its size based on current conditions. As we show later in this paper, the length of this window can greatly affect the QoS. Other interesting work is Chainsaw [12], which uses BT concept but also uses gossip and pushed-based approaches that deviate from the BT mechanisms.

Other approaches, like [13] [14] [15], try to use BitTorrent like technology in order to incorporate streaming capability to the protocol. However the model used, as stated in [16], is totally different, because the clients generally just retransmit the feed they are receiving from an upstream server.

IV. BITTORRENT LIMITATIONS IN STREAMING

In this section, we identify the limitations of BT in providing streaming services and describe how streaming is possible in BitTorrent. Initially we present what kind of time sensitive traffic can be supported by the BitTorrent protocol. We find that BT can potentially deliver Video Streaming services, as long as some minor changes are made to the protocol. In particular, we identify that the Piece selection mechanism of BT is the only module that needs modification.

A. BT vs Time Sensitive Data

In Section II, we have seen that one of the two important mechanisms of BT is the Piece Selection mechanism. Although this mechanism is very efficient in minimizing the probability for a certain piece to become extinct and very effective in providing peers with rare pieces that can use in the Tit-for-Tat mechanism (in order to download pieces from other peers), it fails miserably in case of time sensitive traffic. The reason is that with time sensitive data each piece should be received within a certain time limit. After this deadline, the piece is not useful and will be discarded. This factor is not taken into consideration in the original piece selection mechanism of BT and thus it cannot provide time sensitive distribution services, since pieces are requested based on their rareness and not by their deadline. Consequently, the current piece selection mechanism needs modifications in order BT to be able to support this kind of service.

The other vital mechanism of BT is the Incentive mechanism. This mechanism in case of time sensitive data distribution is even more beneficial for the welfare of the swarm. A Free-Rider participating in the distribution of time insensitive data, who contributes none or only a small portion of its upload capacity, would receive a small fraction of download capacity, due to the Tit-for-Tat policy [17]. This wouldn't be a problem,

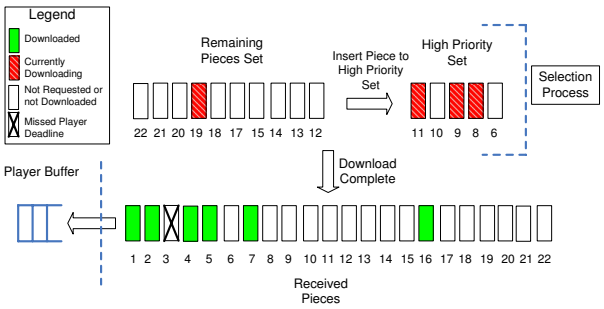

Fig. 1. Our Approach for Supporting Streaming in BT

since the Free-Rider can wait more time until the whole file is downloaded. However, with time sensitive traffic, the Free-Riders cannot afford to wait more time, since each piece has a certain lifetime. In other words, time constrained data distribution provides stronger incentives to peers to avoid being Free-Riders.

B. Time Sensitive Traffic

The most common time sensitive traffic are *Voice*, *Live Video Streaming* and *Video Streaming (playback)*.

Voice and Live Streaming are very demanding in terms of time constrains on delivery and thus increases the difficulty of supporting these services through BT-based protocols. However, the main reason that these services cannot be supported is that BT protocol needs a lot of modifications, which would result in a creation of a new protocol. In more detail, in Voice and Live Video Streaming, packets are not known a priori, but are created dynamically. If we try to translate this property into necessary changes to BT protocol, we would have to totally alter the tracker entity and some parts of the peer communication protocol. This includes functions on how the new pieces are advertised to the peers, how new pieces are updated and how a peer decides which pieces (new or old) to download. Consequently BT cannot support this kind of time sensitive traffic, without replacing the BT protocol with a new protocol.

On the other hand, Playback Video Streaming is a good candidate of time sensitive traffic that can be supported by our approach. In Playback Video Streaming all the pieces of the file are known a priori and therefore the tracker entity can remain the same. The only modification needed is the replacement of the current Piece Selection mechanism, for reasons that were explained in IV-A

V. BITOS

In this Section, we present our approach for providing streaming services in BT. We discuss the main components of the approach and their functionalities, as well as how they can potentially dynamically adjust according to current conditions.

A. Our Approach

Our approach consists of three main components, as shown in Fig. 1. We should point out that Fig. 1 represents a pictorial presentation of how our approach works and does not correspond to the structure of an implementation approach. In detail the functionalities of the components are:

- **Received Pieces**: Contains all the downloaded pieces of the video stream, that the peer has ever downloaded. The state of a piece can be *Downloaded*, *Not-Downloaded* or *Missed*. A piece has state *Missed*, if it didn't meet its deadline to be reproduced by the player.
- **High Priority Set**: Contains the pieces of the Video Stream that have not been *Downloaded* yet, are not *Missed* and are close to be reproduced by the player. Thus, these pieces have higher priority to be requested over the rest of the pieces. This set has a fixed size of pieces and this size is a system parameter. A piece in this set can be in the following states: *Not-Requested* or *Currently-Downloading*.
- **Remaining Pieces Set**: Contains the pieces that have not been *Downloaded*, are not *Missed* and are not in the High Priority Set. A piece can be in the *Not-Requested* or *Currently-Downloading* state.

In the Selection process, the peer chooses with some probability p to download a piece of the video stream, which is contained in the High Priority Set and with probability $1-p$ a piece contained in the Remaining Pieces Set. The probability p represents the balance between the immediate need for a piece and the acquisition of a piece as future "currency[1]". The High Priority Set, contains all the pieces that are quite close to be reproduced. Thus, peer desires to download these pieces earlier, in contrast with the Remaining Pieces Set, which contains pieces that won't be needed in the near future. The probability p, can be adjusted dynamically to adapt to different conditions. The way that this probability is adjusted is explained later. The mechanism used to choose a piece within the High Priority Set or Remaining Pieces Set is the Rarest First mechanism, which is the original mechanism of BT. A minor change of the Rarest First mechanism is that if two or more pieces have the same rareness, the piece which is closer to meet its deadline will be chosen. A peer at any given time can have at maximum a total of k *Currently-Downloading* pieces.

After a piece is downloaded, the piece is removed from its current set and joins the Received Pieces Set (*Download Complete* function in Fig. 1). At the same time, if the piece was in the High Priority Set, the *Insert Piece to High Priority Set* function delivers the next in order piece to the High Priority Set from the Remaining Pieces Set. For example in Fig. 1, piece 12 will move to the High Priority Set if any of the *Currently-Downloading* pieces becomes *Downloaded*. In this way the cardinality of the High Priority Set remains fixed. The pieces within the sets do not have to be sequential since the pieces are not requested in order, i.e pieces 7, 16 are missing from the High Priority Set and Remaining Set respectively since these pieces have been downloaded. We should point out that the Received Pieces Set contains the downloaded pieces that can be shared with other peers.

[1]Pieces from the Remaining Pieces Set are more rare and thus their acquisition is beneficial due to the Tit-for-Tat policy.

Determining the timeliness of the arrival: After the initiation of the player the Player Buffer requests the needed pieces from the Received Pieces Buffer. Another important function of the system, which is not shown in Fig. 1 is the *Piece Deadline* function. This function is responsible for every *Not-Downloaded* or *Currently-Downloading* piece, to determine if the piece can be downloaded on time or not. If the piece cannot meet its playback deadline, then it will not be asked to be downloaded (or its download can be aborted) and will be marked as *Missed*, i.e piece 3 in Fig.1. In order to make this decision, the function compares the expected playback time of the piece and the minimum time[2] needed to download it. If the expected playback time is smaller, then the piece won't arrive on time and consequently won't be needed.

We should point out that the described approach is very simple to implement and can be easily incorporated into BT by just replacing the current Piece Selection mechanism.

B. The Effect of the probability p

The probability p can have an important impact on the performance of the Streaming. Large values of p guarantee that the pieces that will be reproduced soon, will be requested for downloading earlier than the rest of the pieces of the video stream. On the other hand, this could lead to a situation in which the peer chooses to download pieces that most of the peers have. Therefore, the peer wouldn't have any rare pieces to exchange and consequently would be choked by most of the peers according to the Incentive mechanism of BT. Apart from this, rare pieces that are currently available, might not be available in the future. For example, peers that have these pieces might leave the network or fail. Hence, by acquiring these rare pieces before they become extinct we can increase the QoS.

The adaptation of the probability p can be triggered by events, such as a miss of a deadline. For example, a miss of a piece's deadline while there are many pieces unplayed inside the Received Pieces Buffer indicates that the probability p should be increased in order to give higher priority to the pieces that have shorter deadlines. On the other hand, if we miss many deadlines and there are no other pieces inside the Received Pieces Set and the download rate is small, this could indicate that the peer is chocked by most of its peers, because it doesn't have pieces to exchange. Therefore, the decrease of the value of the probability can be helpful in order to acquire some rare pieces that the peer can use as leverage.

VI. EXPERIMENTAL EVALUATION

In order to evaluate our approach we have developed a BitTorrent simulator. In this simulator we have included all the functionalities of the original BT protocol and we have also incorporated the BiToS streaming model.

We evaluate our model by using a synthetic scenario. In this scenario we have 4 seeders and a total of 400 peers arriving in flash crowd, which is a typical behavior in the BitTorrent swarms [9]. In these scenarios (flash crowds), the classic streaming server model performs poorly and the importance of the p2p approaches, which provide a robust and effective solution, is revealed. For streaming, we used a video file of 10 minutes length, which was encoded using quality of $500Kbps$. In order to support the streaming service, the peers should be able to download with rate at least the rate of the stream, otherwise the peer would experience poor streaming quality. For this reason, in our scenario, all peers have total download rate equal to $500Kbps$. The upload rate is also set to $500Kbps$, because according to [10] the download rate is positively correlated to the upload rate. In more detail, Legout et al. [10] showed through experiments that the amount of uploaded data is very close to the downloaded data. This is explained by the Incentive mechanism of BT. Thus, if we want to support a streaming service of the particular quality, the peers should have download/upload rate at least equal to the streaming rate. The rest of the key parameters of the BT protocol, such as the active peer set[3], are set to their default values. Particularly for the active peer set, our decision to retain the default value is strengthened by the fact that, Zhang et al. [11] found that for the same size of active peer set they observed optimal performance.

A. Experimental Results

In the evaluation, we compare the performance of our approach with three different mechanisms in selecting pieces.

- **Sequential** ($p = 1$): The pieces are requested in order within the High Priority Set, without taking into account their rareness. In other words, this mechanism represents how a straight forward streaming would work.
- **Rarest First** ($p = 1$): The pieces are requested only within the High Priority Set, using the Rarest First mechanism.
- **Rarest First** ($p = 0.8$): The pieces are requested with probability 80% within the High Priority Set and with 20% probability within the Remaining Pieces Set, using the Rarest First mechanism.

In the evaluation, we don't do any dynamic adaptation of the probability p, as explained in V-B, in order to analyze the dynamics of the different parameters easier. Note that the original BT behavior corresponds approximately to $p = 1$ and High Priority Set Size equals to 100% of the file.

The main metric for the evaluation of the mechanisms is the playback continuity of the stream. Therefore, we use the *Continuity Index (CI)* metric as defined in [11]. The Continuity Index is defined as the number of pieces that arrived before the playback deadline over the total number of pieces. Fig. 2 shows the CI for the three mechanisms as a function of the size of the High Priority Set.

Our limited rarest first works well for streaming: From Fig. 2, we can clearly see that the rarest first mechanisms behave better than the sequential mechanism. The reason is that the rarest first mechanisms, increase the diversity of the pieces inside the swarm by replicating first the most rare pieces. Thus, it increases the parallelism in the downloading process and

[2]Here we use a lower bound of the expected download time which we define as the remaining of the piece divided by link bandwidth.

[3]The active peer set is the maximum number of concurrent upload connections that a peer can have, the default is 4.

utilizes better the bandwidth within the swarm. However, in the sequential mechanism the same pieces are requested by all peers and consequently there are only few providers of these pieces, which results in low replication rate.

Selecting the size of the High Priority Set: Another interesting observation from Fig. 2 is that in the rarest first mechanisms ($p = 1, p = 0.8$), for small ($< 5\%$) or large size ($> 20\%$) of the High Priority Set the CI is decreased. The reason is that for small size ($< 5\%$) of the High Priority Set, peers do not increase the diversity of the pieces because they tend to download the same pieces due to the small size of the set. This results in low use of parallelism in downloading, which stalls the downloading process and results in low CI. On the other hand when the size of the list is large ($> 20\%$), the peer downloads pieces based on their rareness, without considering their deadline and thus the CI drops. In other words, the optimal size of the High Priority Set ($\simeq 8\%$) must capture the pieces that will be needed soon for the playback and at the same time is large enough for the rarest first piece selection mechanism to work properly.

The effect of the probability p: The effect of p on the performance of the Rarest First mechanism is obvious. In Fig. 2, it is clear that the Rarest First with probability $p = 0.8$ performs better, for small reasonable sizes of the High Priority Set. The reason is that: a) it acquires some rare pieces before they become extinct and b) it increases the diversity of the exchanged pieces between peers. Thus the CI is improved. However, with larger sizes of the High Priority Set, the pieces inside the list are already far away from playback time and therefore retrieving pieces from outside the list (with 20% probability) degrades the overall performance of the system even more. We should note that the optimal value of the probability for the Rarest First mechanism highly depends on the dynamics of the scenario. Thus, for our specific scenario[4] we tried different values for the probability and we found that for probability $p = 0.8$, we get the best results.

In Fig. 2, we can see that the CI is not getting much worse as the size of the High Priority Set is increased over 30% of the file. This is a consequence of the flash crowd scenario which we use in our simulation, in which all peers arrive almost simultaneously. Thus, peers have similar playback times and therefore the pieces required by each peer are almost the same. This fact together with our modification of the rarest first algorithm (the piece with shortest deadline is chosen among pieces with the same rareness), explain the almost flat line after the size of the High Priority Set exceeds 30% of the file.

Robustness to greedy peer behavior: In order to illustrate the preponderance of the Rarest First mechanism compared to the Sequential, we have created a flash crowd scenario in which the size of the High Priority Set, file size and arrival patterns of peers are constant and we stream a 5 min Video-file. By varying the seeding time[5] the dynamics of the system change and therefore we can observe the robustness of the two methods.

[4]In a different scenario, the optimal operation point may be different. This suggests the need for an adaptive mechanism to set the value of p. Although we have some preliminary ideas, this extends beyond the scope of this work.

[5]The amount of time that a peer stays in the network after it becomes a seeder.

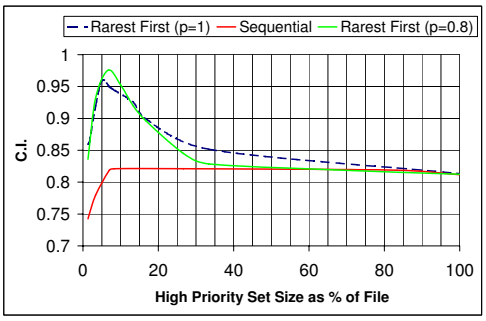

Fig. 2. High Priority Set Size as a percentage of File

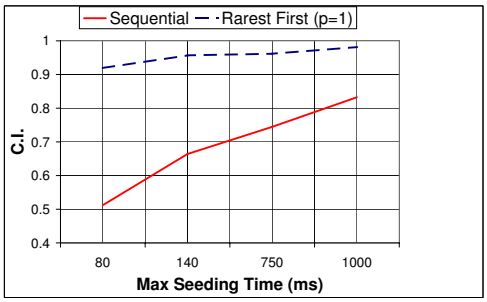

Fig. 3. C.I. versus the MAX Seeding time

In Fig. 3, it is obvious that in a highly dynamic environment with peers leaving as soon as they download the file, the rarest first algorithm is much more robust compared to the sequential. This holds because, with the rarest first algorithm each peer chooses to download rare pieces which might not be available in the future. Pieces might not be available due to the departure of the peer that is offering it.

Note also that the buffering time has a significant effect on the performance. Large buffering time, clearly will increase the performance of the protocol.

VII. CONCLUSION

In this work we have shown that Streaming in BT is possible under our proposed approach. We have shown through simulations that our approach is feasible and can be easily (with minor modifications) incorporated into the original BT protocol.

As future work we aim at investigating the dynamics of the seeders/leechers ratio, as well as the relation between them. Moreover, we want to explore and identify the events or conditions that can trigger a dynamic adaptation of the probability p and the Desired Pieces list size. Such a dynamic scheme would be more robust in environment changes and can improve the streaming performance even more. Finally, we plan to incorporate our modifications into the BT Mainline client [18] and evaluate our model in PlanetLab [19], in order to further investigate the effectiveness and the robustness of our streaming model in a real network deployment.

REFERENCES

[1] B. Cohen. Incentives build robustness in bittorrent. In *1st Workshop on the Economics of Peer-2-Peer Systems*, Berkley, CA, June 5-6 2003.

[2] CacheLogic. http://www.cachelogic.com/research/slide1.php.
[3] C. Dana, D. Li, D. Harrison, and C. Chuah. Bass: Bittorrent assisted streaming system for video-on-demand. In *International Workshop on Multimedia Signal Processing(MMsP) IEEE Press*, 2005.
[4] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 68–77, New York, NY, USA, 2005. ACM Press.
[5] BitTorrent Specifications. http://wiki.theory.org/BitTorrentSpecification.
[6] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on cooperation in bittorrent communities. In *Proceedings of ACM Sigcomm*, Philadelphia, PA, Aug 2005.
[7] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *Proceedings of ACM Sigcomm*, Philadelphia, PA, Aug 2005.
[8] JA. Pouwelse, P. Garbacki, D.H.J Epema, and HJ. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of IPTPS*, Ithaca, New York, Feb 2005.
[9] M. Izal, G. Urvoy-Keller, P.A. Felber E.W. Biersack, A. Al Hamra, and L. Garc'es-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *Proceedings of PAM*, Antibes Juan-les-Pins, France, Apr 2004.
[10] A. Legout and G. Urvoy-Kellerand P. Michiardi. Understanding bittorrent: An experimental perspective. Technical report, Sophia Antipolis, France, 2005.
[11] X. Zhang, J. Liu, B. Li, and T.P. Yum. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of IEEE/INFOCOM*, Miami, March 2005.
[12] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A.E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of IPTPS*, Ithaca, New York, Feb 2005.
[13] PeerCast. http://www.peercast.org/.
[14] Streamer P2P. http://www.streamerp2p.com/.
[15] P2P-Radio. http://p2p-radio.sourceforge.net/.
[16] Bit Torrent FAQ. http://wiki.theory.org/BitTorrentFAQ.
[17] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of ACM Sigcomm*, Portland, OR, Aug 2004.
[18] The Official BitTorrent Home Page. http://www.bittorrent.com/.
[19] The PlanetLab project. http://www.planet-lab.org/.