"Take One Get One Free": Leveraging P2P Networks for Content Promotion

Himabindu Pucha, Sabyasachi Roy and Y. Charlie Hu
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907

***Abstract*— The nature of digital content is undergoing a radical transformation due to the growing infusion of user-generated content. Users that generate content have a strong motivation to actively promote their content in order to achieve publicity, recognition or to simply spread their viewpoints, knowledge and creations. Existing content sharing mechanisms available to such users, such as in p2p file sharing, are pull-based, relegating the user to a passive role. To enable active participation from users, we propose Promoter, a framework that enhances p2p file sharing systems to support content promotion in addition to the available pull-based content search. Promotion is enabled by publicizing a user's content to peers and disseminating the content by incentivizing peers to download and further propagate it.**

I. Introduction

The diversity and evolution of digital content is at the heart of the tremendous growth the Internet has witnessed. Traditionally, digital content is primarily produced by commercial web sites to be downloaded by users. However, the nature of digital content is currently undergoing a radical transformation. The advent of cheap and easy to use digital media technology and computing is causing an explosion of user-generated content such as home-made movies, documentaries, pictures, op-eds, amateur music audio/video or even software. The increase in broadband connectivity enables distribution of this content to other users. For example, the new-found popularity and growth of user-generated content is evident from the significant amount of such content in the programming of a new TV network, Current [18], reaching over 20 million homes. Another example is the wide-spread adoption of the "blogging" phenomenon. Recent data [16] show that the number of articles in just one "blog-space" provider grew from 0 to 600 million from July 2003-July 2005 and the rate of user-generated content is growing at 2 million articles every day. Similarly, the sharing of digital photos at Flickr.com is growing at 5-10% a week and 60,000 photos are added a day. MySpace.com, with 25 million members posting videos and photos, is yet another example of the growth in user-generated content.

This shift in the nature of digital content has important implications. Particularly, when users generate content, they have a strong motivation to promote their content in order to achieve publicity, recognition, and monetary gain, or to simply spread their viewpoints, knowledge, and creations. The desire to promote content could also be motivated by the passion that people harbor about issues of importance, say, in a political, religious or a cultural arena. The phenomenal success of blogs indicates the presence of such passion that motivates users to promote content created by or obtained from other peers in addition to their own content. Thus, we expect that the growing trend of user-generated content will result in users desiring to play an *active* role in the dissemination of their content by informing other users about its existence as well as recruiting them to download it.

Currently, digital content is shared using either the World Wide Web or a peer-to-peer file sharing network like Gnutella [21], Napster [20], Kazaa [19]. However, in both the WWW and p2p dissemination models, the content-keeper is relegated to a *passive* role waiting for the content-seeker to discover its content. Such pull-based means do not support active promotion of content as an integral part of content sharing. Presently, users can promote their content by other offline means such as advertising through news groups, blogs, personal web sites, personal RSS feeds [12] or through word-of-mouth propagation. However, this offline method of informing content-seekers is cumbersome, slow and relies on the fact that other users will notice the content advertisement. Further, alternative means such as advertising in print, radio, or television media or through other popular web sites is expensive and not feasible for most users.

This position paper advocates an active online push-based mechanism to promote content as a potentially useful feature that complements the existing pull-based content sharing. P2p networks provide a natural platform to implement such a system since the large pre-existing virtual social network can be leveraged for promotion of digital content. To this end, we propose *Promoter*, a framework that enhances any p2p pull-based file sharing protocol to support active promotion of content. Specifically, Promoter **publicizes** the existence of content among other users of a file-sharing community, and attempts to **disseminate** the content to as many peers as possible. Promoter achieves

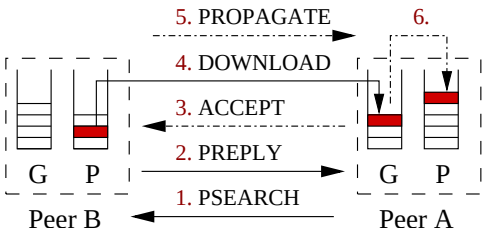


Fig. 1. Promoter overview.

these objectives with a promotion process that does not adversely affect existing peers. We believe that systems such as Promoter can enable p2p networks to become the method of choice for delivering the next generation of digital content.

II. Promoter

Promoter builds upon any existing pull-based p2p file sharing protocol that provides a *search* primitive to locate content and a *reply* primitive that provides a set of content items matching the search. Promoter retains these primitives to provide pull-based search. Further, Promoter enables content promotion by providing two new primitives that publicize and disseminate the content: *psearch* and *preply*. These new messages are propagated similarly to *search/reply* messages but have a modified payload. For explanation, let A be a peer issuing the search K consisting of n keywords ($K = \{k_1, k_2, ...k_n\}$) and B be a peer receiving the search.

Publicizing Promoter peers separate their content into two separate bins (Figure 1) of *promote-worthy* content (P) and general content (G). A Promoter peer publicizes its content items as follows: (1) When A is interested in learning about promoted content, it issues a *psearch* for K with payload identical to *search*, except for a flag indicating its interest. (2) When B receives a *psearch* for which it does not have matching content, or when B does not have any promote-worthy content, or when the content matching K is part of P in B, the behavior of B is similar to when it receives a basic *search* message and hence no publicizing is attempted. (3) However, when the content matching K is in G, B generates a *preply* message. In addition to the search hit information, *preply* piggybacks information about content from P. Each search hit is accompanied with exactly one promoted item information. (4) A is now informed of the promote-worthy content of all the peers that have a hit for K. The promoted content and its description can be viewed via a GUI. Peers can disable promotions by simply using the basic *search* message. Thus, Promoter is not overly aggressive in that it does not generate extra messages to publicize content and does not send unwanted content promotions.

Disseminating The second objective of Promoter is to disseminate the promote-worthy content to peers. This can be achieved when (1) the peers who learn about the promote-worthy content decide to download the content, (2) the peers who download the promote-worthy content offer it to other peers when they seek such content, and (3) the peers who download the promote-worthy content actively promote this acquired content themselves. Peers could be motivated [7] to download or promote content by intrinsic factors such as altruism, reputation or interpersonal factors such as affiliation or liking. Although the promoting peer cannot control the above factors, it can control extrinsic factors such as *rewards* that can also motivate other peers to download and/or further promote the content. Thus, Promoter attempts to spread the promote-worthy content by **incentivizing** peers.

B disseminates its promote-worthy content as follows: (1) Similar to as in publicizing, when B receives a *psearch* which results in no hits or when B has no promote-worthy content, B does not attempt to disseminate any information. (2) When B has a hit in P in response to a *psearch* from A, it piggybacks a *download incentive* offer in the *preply* that rewards A if it chooses B's content over other peers. (3) On the other hand, when B has a hit in G, B piggybacks a "take one get one free" offer along with a download incentive to A. This offer rewards A if it downloads the content it desires along with an associated promote-worthy content item publicized by B. (4) A sends an *accept* message (Figure 1) to notify B that it accepts B's offer. (5) A now downloads its desired content *followed* by the promoted content and collects the associated reward from B. Note that A can fetch the content item from G in B without accepting offers.

As shown in Figure 1, the download incentive allows a promoted content item to be disseminated to the G bin of other peers thereby increasing the availability of that item and the likelihood that other peers find it when they search for related content. However, the dissemination of the promoted content item can be further enhanced if A moves it from its G to P bin and consequently promotes it further. The user on A may do this for intrinsic reasons if he/she is also passionate about that content. However, Promoter also allows B to motivate A to promote the item from P in B by providing A with *propagate incentive* offers via the *propagate* message (Figure 1) after it completes the download of the content from B. The associated rewards for the propagate incentive are collected when A provides proof to B that it propagated the content further to a new peer. The design of Promoter incentives is described next.

A. Design of Incentives

Central to Promoter are the incentive mechanisms required to motivate peers to *download* and *propagate* promoted content. Currently, p2p systems incentivize peers to contribute upload bandwidth to the system in order to prevent free-riding problems. The incentives range from

increase in reputation or trust [14], [1] resulting in obtaining bandwidth from anyone in the system to simply obtaining bandwidth from the peer to whom you uploaded in the past (tit-for-tat as in BitTorrent [17]). However, incentives required by Promoter differ as follows: (1) In the above system, the incentives are designed such that the peers need to collect incentives to *survive* in the system. Hence, the incentives *enforce* a certain behavior on the users. In contrast, in Promoter, accepting incentives offered by a promoting peer is not essential for a user's survival as the content items can still be obtained without any incentive from the same or a different peer. Hence, Promoter incentives can only be designed to *motivate* peers to download content (not force them). (2) Incentives such as reputation are typically system-wide and long term, i.e., a peer can encash its reputation over time from any other peer in the system. In contrast, Promoter incentives are local (between pairs of peers), i.e., on agreeing to download a promoted object from another peer, a peer gets incentives only from that peer. Thus, we need to revisit the design of incentive mechanisms under the context of the Promoter framework.

Incentives can be loosely classified into offline and online incentives. Offline incentives include monetary compensation, public recognition and fame. Online incentives include incentives offering: (1) High QoS (e.g., improved bandwidth or download time) for user download. QoS incentives are generic as they are applicable to any content being served. (2) Commercial content free of charge when allowed. (3) High quality of content such as audio/video files. (4) Assurance about the authenticity of content. This incentive is attractive due to the increased pollution in file-sharing systems [14]. (5) Gain in reputation or score. In summary, offline incentives are difficult and comparatively slow to offer, obtain or verify. In contrast, online incentives can be easily integrated into system operation and are instantaneous in effect. Further, their authenticity is easily and quickly verifiable.

Promoter requires a *download* and a *propagate* incentive. Download incentive motivates the user to download a promoted item and hence should have immediate gratification. Further, it should be generic and easily encashable from the promoting peer. We thus choose the online incentive of offering high QoS as a suitable download incentive. Specifically, the promoting peer offers high QoS by *reserving an advertised amount of upload bandwidth from its end* for the receiving peer throughout the download of the queried content and the associated promoted item. Note that downloading an additional promoted item is not a burden for the users in a p2p file sharing network. Unlike the Web which is an interactive system, p2p file sharing networks are batch-mode delivery systems [4]. Thus, while Web users generate a steady stream of requests requiring instant gratification, p2p users have potentially higher inter-arrival time between consecutive file downloads.

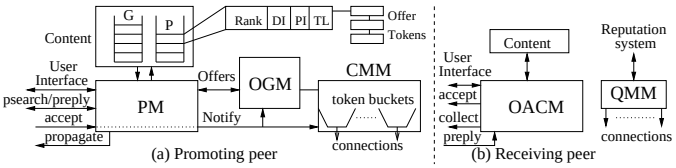


Fig. 2. Architectural components at promoting/receiving peers.

On the other hand, the propagate incentive, by its very nature, has delayed gratification, i.e., the receiving peer that downloads the promoted content can disseminate the content to other peers only when it participates in the next search and can then collect its reward. Thus, the propagate incentive can be offered using either offline or online schemes. Potential choices for the propagate incentive include QoS offers on future downloads or mutual agreement on distributing a peer's promoted objects (tit-for-tat). For the scope of this paper, we offer a bandwidth reservation for one future download of the peer's choice as a propagate incentive.

B. Architectural Components

This section describes the components of the Promoter framework. The promoting peer requires the *Publicizer*, *Offer Generator*, and *Connection Manager* modules. The receiving peer requires the *Offer Acceptor and Collector*, and *QoS Monitor* modules. Figure 2 depicts the different components and the interaction between them.

Publicizer Module (PM): The PM exports an interface for the user to separate shared content into P (promote-worthy) and G (general) categories. Each item I in P has the following associated metadata (Figure 2(a)): (1) A user-assigned rank, (2) boolean values to specify whether download incentives (DI) and/or propagate incentives (PI) will be provided for I, and (3) a token list (TL) (explained later).

The PM operates as follows: It searches for keywords received in a *psearch* in both P and G. In the event of a hit for a content item H belonging to P, if the DI flag is set, the PM piggybacks a download incentive offer obtained from the OGM (Offer Generator Module discussed below) in the *preply* message delivered to the querying peer. In the event of a hit for a content item H in G, the PM first picks an item I from P to promote such that the number of times an item is chosen is proportional to its rank. We also ensure that the size of I does not exceed that of H in order to be effective, e.g., it is difficult to interest a user to download an extra 5MB of promoted content when the content it desires is only 500KB. This process is repeated for each content item H that is a hit for the keywords in the *psearch*. These item pairs are then included in *preply* along with a description of each I. The PM requests the OGM for a bandwidth reservation for this *preply*. The PM

then awaits an *accept* and conveys its receipt to the CMM (Connection Manager Module) which enforces the offer and the OGM which keeps track of pending offers. The PM also sends appropriate propagate incentives to the peers that download a promoted item. Thus, the PM handles all the protocol messages and communicates with other modules to enable content promotion. Note that if the OGM is unable to generate an incentive offer for a given item pair due to temporary non-availability of resources, the item pair is sent to the querying peer without an offer.

Offer Generator Module (OGM): The OGM provides a download incentive offering bandwidth reservation to the PM to be sent to the querying peer. At bootstrap, the user sets aside B, a configurable fraction of the total upload bandwidth, for making offers and the OGM dynamically manages B. An offer essentially reserves a fraction of B for a peer. A trivial method of providing offers is to reserve the entire B for a single offer and reuse B when the offer expires or is completed. However, this may lead to underutilization of bandwidth during the entire download duration of the offer when the TCP throughput of the single connection in the offer is less than B due to querying peer download bandwidth constraints or network properties.

There are three potential techniques to counter underutilization of reserved bandwidth: (1) If the TCP throughput is limited by the RTT or the window size and not by the access link of the downloading peer, then parallel downloads for different byte ranges of the same content can be used to saturate B. However, if the limitation is due to congestion along the path, then parallel downloads will not reduce underutilization. (2) Throughput prediction [5] can also be used to generate offers. The achievable throughputs of all the current connections are predicted and summed to obtain current bandwidth usage X. The surplus $B - X$ is then used to make a future offer. (3) Another candidate is a monitoring-based scheme that measures the effective usage of bandwidth. The throughputs of all the current connections are monitored instead of prediction and their steady state values are assumed to be their achievable throughputs. The sum of these values gives the current usage estimate Y. The surplus $B - Y$ is used to make future offers. There is a need to provide extra bandwidth to absorb fluctuations in peer throughput and avoid violation of offers.

The OGM also generates propagate incentives for the PM. Each incentive provides one high QoS download to the peer in exchange for uploading the promoted item to another peer. Each such incentive has a unique offer id that the peer uses to collect the reward in the future. Note that when a peer generates propagate incentives, it needs to monitor its commitments and the claim rate of the awarded peers to maintain minimum guarantees of QoS. In the above schemes, each peer advertises the best offer it can provide at that instant. Alternatively, a peer can generate *competitive* offers as in bidding. The peer can use *psearch* messages to snoop on offers generated by other promoters in the network and offer just enough to outbid them.

Connection Manager Module (CMM): In addition to aiding the OGM in thwarting underutilization, the CMM enforces the QoS offers made to peers. The CMM reserves the amount of bandwidth offered to a peer by allocating a token bucket for each connection. The rate R and depth B [2] of this token bucket is obtained using the fluid model approximation [9], [8]. The token bucket is bootstrapped with a value equal to the offer made and subsequently adjusted based on the countermeasure for underutilization.

Offer Acceptor and Collector Module (OACM): Once all the replies are received by a querying peer, it has several offers to choose from. The offer selection could be done by the user or automated via the OACM as follows: One possible way is to take a greedy approach by choosing the offer with the highest reserved bandwidth. The querying peer can also pick the peer that is predicted to provide the highest throughput. Such prediction also considers the reserved bandwidth in the offer. A peer could also accept offers from multiple peers and download byte ranges of the content from them.

In contrast, the propagate incentive offers need to be accepted by the user. The OACM stores the list of accepted propagate offer tokens in the TL metadata field (Figure 2(a)) for each item I in P. Each offer token contains the offering peer's IP address and an offer id. Figure 3 shows how the OACM collects the reward associated with propagate incentives. Assume that A downloaded I from B and received propagate offer tokens from B. If A uploads I to another peer C (step 2), A's OACM can collect a reward from B. A provides C's IP, the propagated item (I) and an associated offer token in a *collect* message (step 3) to B so that B can verify the upload to C (step 4) and provide the reward to A (step 5). B verifies the propagation by asking C to send the hash of a nonce and a random set of file blocks to B so that B can compare the hash to its own result. It is important to discourage collusion among malicious peers. For example, the random set of file blocks and nonce whose hash value is requested for verification needs to be different for A and C and there should be a reply timeout set to discourage C from getting the hash values from A without downloading the file. Without such mechanisms, A could collude with C and collect a reward from B without C having downloaded the promoted item. Note that the above scheme is just one design choice. Incorporating detailed security schemes is out of the scope of this position paper and part of our future work.

Note that A can even choose to provide its own propagate offer tokens for promoting I to C. Thus when C promotes the item, C can collect a reward from A and A can also

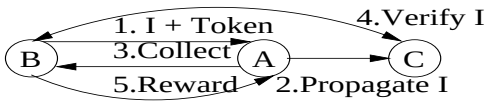


Fig. 3. How propagate incentives work.

collect a reward from B.

QoS Monitor Module (QMM): The QMM monitors the QoS offered connections to check if the promoter is respecting the offers made or not. Note that when an offer is flagged as violated, the QMM needs to ensure that the limitation is not on its end either through previous estimates of achievable throughput or by measuring network properties [5].

III. Preliminary Results and Status

To understand Promoter dynamics, we implemented Promoter with Gnutella, **Promutella**, in a trace-driven simulator. We collected a 2-day Gnutella trace using 4 modified Gtk-Gnutella clients. The trace was anonymized and analyzed to extract queries with keywords and a file list for each peer. The trace contains 1540 peers, 8019 files, and 26959 keywords. In the simulation, each file is placed at the node it is first seen in a QueryHit. Each node is assigned an upload bandwidth randomly chosen between 300-600 Kbps similar to in [6]. In our simulation, each querying peer issues a PQUERY with a TTL of 4 that is flooded via its 4 randomly chosen neighbors. Every peer is assumed to have promote-worthy content and piggybacks one such item for every file returned in a PQUERY HIT. The peer offers to reserve 70% of its upload bandwidth for the duration of the transfer as a download incentive. The querying peer greedily chooses the content item with the highest reserved bandwidth and the associated promoted item to download. The file sizes of all the files are randomly chosen between 1MB and 10MB as a majority of p2p requests are in this range [4]. Propagate incentives are not simulated in these preliminary results.

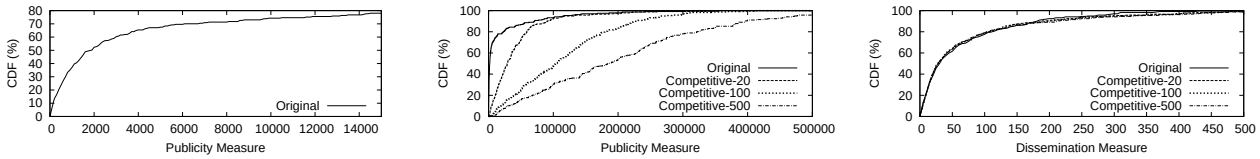
Our simulations evaluate: (1) Publicity Measure (PM): number of times a node is able to publicize its content. (2) Dissemination Measure (DM): number of times a node is able to disseminate promote-worthy content. PM/node is directly related to the number of queries/node. This in turn depends on the total number of queries in the system (20,000) and the connectivity of the querying nodes. PM/node also depends on the popularity of files at the nodes receiving the query. Figure 4(a) shows the CDF of PM/node. It shows that, in 44 hours, 90% of the peers can publicize their content more than 500 times, and 20% of nodes that either have many files to share or received many queries were able to publicize their content more than 15,000 times. To remove the disparity between peers with respect to availability of files, we also simulate *competitive publicizing*, in which peers monitor forwarded protocol messages and crawl k popular files. Since such popular files are more likely to be downloaded by other peers, they increase the chances for a peer to promote its content. k is varied as 20, 100 and 500. Figure 4(b) shows that competitive publicizing greatly improves the PM/node. For instance, for $k = 20$, 80% of peers have a PM of more than 20,000. Increasing k further increases PM/node.

DM/node depends on its PM and its upload bandwidth compared to other peers responding to a query. Also, the sum of DM/node for all nodes is 20,000, the total number of queries in the system, as only one promoted item is downloaded per query. Figure 4(c) shows that 80% of the peers upload promoted items more than 20 times while 30% of the peers upload promoted items more than 100 times. Thus, simulations show that Promutella helps promote the content for a large fraction of peers. Note that these results are conservative since they ignore the downloads that may occur due to interest generated from publicizing as well as from propagate incentives.

Development Status: We are developing a prototype implementation of Promutella by modifying the LimeWire Gnutella client. Promutella clients implement the basic Gnutella protocol and thus inter-operate with vanilla Gnutella clients, thereby allowing an incremental deployment. A Promutella client A that wishes to receive promotion offers sends out a PQUERY which is a Gnutella QUERY with a special descriptor id in which bytes 12-16 have a pattern [1010...]. Such queries are forwarded transparently by non-Promutella peers to reach Promutella-peers. Non-Promutella peers respond to the queries with non-promotional query hits. On the other hand, a Promutella peer B that receives the PQUERY can identify (using the descriptor id) that the query originated from a Promutella peer and thus generates offers. Offers are inserted in the trailer [21] of the extended QUERY HIT messages. An offer consists of an offer id, a bandwidth reservation and the promoted content to be downloaded to satisfy the offer. The QUERY HIT is forwarded back to A as in Gnutella since the peers in between are not affected by the piggybacked information. A then performs a HTTP GET with an Offer-Accept header specifying the offer id it has accepted. If no Offer-Accept header is sent by A, the original content searched for by A is returned without any QoS. Currently, we are experimenting with a prototype deployment of Promoter using ModelNet [13] and plan to perform wide area tests using PlanetLab [11].

IV. Discussion

Free riding and Promoting: P2p network performance depends on the number of peers hosting content and their upload speeds. The large fraction of free-riders observed in current p2p networks hurts performance. It can be argued that although free-riders may not allow uploads of general content, it is likely that a fraction of free-riders will have a



(a) (b) (c)

Fig. 4. Trace-driven evaluation of Promoter behavior.

passion for promoting their own content. In this scenario, Promoter may reduce free-riding by forcing such users to upload general content in order to promote their own. Alternatively, Promoter may cause some peers to reserve their entire upload bandwidth for promotions only, resulting in a system where content is typically available along with promoted content.

Security: To discourage malicious behavior, Promoter requires a reputation system. A promoting peer may fail to honor its incentive offers or promote misrepresented content. Alternatively, a receiving peer may use up the QoS for its queried item and subsequently refuse to download the promoted item. Such peers need to be blacklisted so that no offers are accepted from or made to them. On the other hand, peers with a good reputation may be favored while generating offers.

Promoter and Spam: The push-based nature of Promoter raises spamming concerns [3]. However, Promoter does not encourage spamming due to the following reasons: (1) Promoter is different from pushing in email since users have personal gain in receiving pushed content and explicitly choose to do so. (2) Unlike in email spamming, peers in Promoter pay a price of providing good QoS downloads of content that a *user does want* in order to be able to "push" their content. (3) Even if the promoted content is misrepresented and can be considered as spam, the reputation system in place will automatically blacklist such a promoter. Thus there is both recipient control over the promoted content received as well as a deterrent and cost for the promoter in promoting fraud content, both of which are absent in email.

Commercializing Promoter: Promoter can be leveraged as an advertising tool by commercial operators. For example, media companies can promote movie trailers through their own distributed promoter clients (e.g. run on commercial slices on PlanetLab [10]) over various file-sharing networks, by offering as bait other high bandwidth downloads users desire. This is beneficial to users as it rewards them for viewing advertisements. Incorporating promotion of commercial content (e.g., copyrighted music) has different challenges and is part of our future work. We currently envision Promoter as a tool for promotion of user-generated free content.

Additional Features: We are working on including two additional features in Promoter to further enhance its usability: (1) A *promotion filter* (similar to a spam-filter) is used by each peer node to filter out unwanted content, and (2) The keywords in the search message itself (which indicate peer interests) are used to pick an item to be promoted. Part of our future work is the inclusion of *Interest-based promotion*. This technique uses a DHT index [15] to store each peer's interests which is then queried to drive the selection of content items to be promoted to a peer.

REFERENCES

[1] E. Damiani, D. Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of ACM CCS*, 2002.
[2] K. Dovrolis, M. Vedarn, and P. Ramanathan. The Selection of the Token Bucket Parameters in the IETF Guaranteed Service Class. Technical report, Department of ECE, University of Wisconsin-Madison, November 1997.
[3] Z. Duan, K. Gopalan, and Y. Dong. Push vs. pull: Implications of protocol design on controlling unwanted traffi c. In *Proc. of USENIX SRUTI*, 2005.
[4] 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 *Proc of ACM SOSP*, 2003.
[5] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer tcp throughput. In *Proc. of ACM SIGCOMM*, 2005.
[6] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proc. of ACM SOSP*, 2003.
[7] S. M. Lui, K. R. Lang, and S. H. Kwok. Participation incentive mechanisms in peer-to-peer subscription systems. In *Proc. of HICSS*, 2002.
[8] A. Parekh and R. Gallagher. A generalized processor sharing approach to fbw control-the multiple node case. *IEEE/ACM Trans. Netw.*, 2(2), 1993.
[9] A. Parekh and R. Gallagher. A generalized processor sharing approach to fbw control-the single node case. *IEEE/ACM Trans. Netw.*, 1(3), 1993.
[10] L. Peterson. Commercialization of PlanetLab: A Whitepaper. Technical Report PDN–05–027, PlanetLab Consortium, June 2005.
[11] PlanetLab. http://www.planet-lab.org.
[12] D. Sandler, A. Mislove, A. Post, and P. Druschel. FeedTree: Sharing Web Micronews with Peer-to-Peer Event Notifi cation. In *Proc. of IPTPS*, 2005.
[13] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of USENIX OSDI*, December 2002.
[14] K. Walsh and E. G.Sirer. Fighting Peer-to-Peer SPAM and Decoys with Object Reputation. In *Proc. of p2pecon*, 2005.
[15] R. Zhang and Y. C. Hu. Assisted Peer-to-Peer Search with Partial Indexing. In *Proc. of IEEE INFOCOM*, 2005.
[16] Ask jeeves blog. http://blog.ask.com/2005/07/what_feeds_matt.html.
[17] Bittorrent. http://bitconjurer.org/BitTorrent.
[18] Current tv. http://www.current.tv.
[19] Kazaa. http://www.kazaa.com.
[20] Napster. http://www.napster.com/.
[21] The Gnutella protocol specifi cation, 2000. http://dss.clip2.com/GnutellaProtocol04.pdf.