Combating Hidden Action in Unstructured Peer-to-Peer Systems

Qi Zhao, Jianzhong Zhang, and Jingdong Xu

Abstract—**In unstructured peer-to-peer systems, cooperation by the intermediate peers are essential for the success of queries. However, intermediate peers may choose to forward packets at a low priority or not forward the packets at all, which is referred as peers' hidden action. Hidden Action may lead to significant decrement of search efficiency. In contrast to building a global system with reputations or economics, we proposed MSSF, an improved search method, to help queries route around the peers with hidden action. MSSF does not need to check other peers' behavior. It automatically adapts to change query routes according to the previous query results. Simulation results show that MSSF is more robust than Gnutella flooding when peers with hidden action increase.**

Index Terms—**Hidden Action, Peer-to-Peer, Unstructured**

I. Introduction

Peer-to-Peer (P2P) systems provide a good substrate for building large scale data sharing and content distribution applications. The most popular P2P applications in today's Internet, like Gnutella [1] and Kazaa [2], are unstructured. In these systems, when a source peer needs to query an object, it broadcasts a query to its neighbors. If a peer receiving the query finds the TTL is not zero, it will forward the query to its neighbors, and so forth. If the peer receiving the query can provide the requested object, a response message will be sent back to the source peer along the inverse of the query path.

Most Peer-to-Peer (P2P) file-sharing systems assume that all peers are cooperating for the benefit of the community. However, there is a significant portion of peers who leech resources from the system without contributing any in return. As a result, in unstructured P2P file-sharing systems, intermediate peers may choose to forward packets at a low priority or not forward the packets at all. We call this behavior hidden action.

Hidden Action may lead to decrement of search efficiency. The source peers can provide incentives, e.g., in the form of payments, to encourage the intermediate peers to forward their queries. However, the actions of the intermediate peers are often hidden from the source peers. In many cases, the source peers can only observe whether or not the query is successful, and cannot attribute failure to a specific peer on the path. Even if some form of monitoring mechanism allows them to pinpoint the location of the failure, they may still be unable to attribute the cause of failure to either the deliberate action of the intermediate peer, or to some external factors beyond the control of the intermediate peer, such as network congestion, channel interference, or data corruption.

Aiming at improving the search efficiency, we propose MSSF. It is an efficient search mechanism using Multiple Small-Scale Floods for each query instead of a flood with a large TTL. The MSSF mechanism tries to route queries to the honest nodes, which are not necessarily within a certain radius from the source peer. We hope MSSF can overcome the efficiency decrement caused by hidden action. We also hope MSSF can perform better than traditional flooding based search in a network where hidden action is rare.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents our implementation of MSSF mechanism in details. In Section 4, the simulation results are presented, followed by conclusions in Section 5.

II. Related Work

The study of non-cooperative behavior in communication networks, and the design of incentives, has received significant attention. [3] considers the problem of malicious behavior, where peers respond positively to route requests but then fail to forward the actual packets. It proposes to mitigate it by detection and report mechanisms that will essentially help to route around the malicious peers. In order to mitigate selfish behavior, some approaches [4, 5, 6] require reputation exchange between peers, or simply first-hand observations [7]. Other approaches propose payment schemes [8] to encourage cooperation. In contrast to building a global system with reputations or economics that requires users to cooperate or some central authorities, [9] proposes a simple alternative solution. The amount of service a peer provides a neighbor is related to the service it receives from that neighbor.

The problem of information asymmetry and hidden-action (also known as moral hazard) is well studied in the economics literature [10, 11, 12]. [10] identifies the problem of moral hazard in production teams, and shows that it is impossible to design a sharing rule which is efficient and budget-balanced.

Qi Zhao is with the Computer Science Department, Nankai University, Tianjin, 300071 China (e-mail: qizhao6688@hotmail.com).

Jianzhong Zhang is with the Computer Science Department, Nankai University, Tianjin, 300071 China (e-mail: zjz@mail.nankai.edu.cn).

Jingdong Xu is with the Computer Science Department, Nankai University, Tianjin, 300071 China (e-mail: xujd@mail.nankai.edu.cn).

1-4244-0463-0/06/$20.00 ©2006 IEEE

[11] shows that this task is made possible when production takes place sequentially, and [12] distinguishes between strategic substitutes and complements, and shows that a principal is better-off under sequential production of strategic complements, but prefers simultaneous production of strategic substitutes.

Many search schemes in unstructured P2P networks have been proposed in the last few years. Ref. [13] presents a thorough categorization and description of many approaches. A non-forwarding search protocol, named GUESS, was presented in [14]. The basic architecture of MSSF partly draws inspiration from GUESS. Under the GUESS protocol, a query is conducted by iteratively contacting different neighbors until a number of objects are found. The querying source has full control over query process, thus no network resource is wasted to search for popular objects. However, to find rare objects, the source peer must wait a long time to receive sufficient results. To reduce the use of Gnutella flooding, Interest-Based Locality [15] enables a peer to create shortcut links with those peers serving it qualified results previously. Therefore, peers that are not necessarily neighboring ones can communicate directly. Our MSSF mechanism also draws inspiration from this idea. However, Interest-Based Locality does not provide a mechanism to focus query propagation beyond the first hop.

III. MULTIPLE SMALL SCALE FLOODS

To illustrate the basic ideas of our method, let us begin with a simple example as shown in Figure 1 and Figure 2.

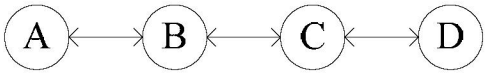

Fig. 1. Current Search Method.

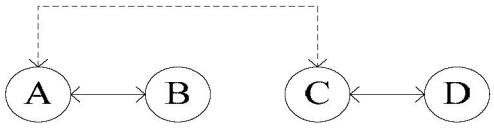

Fig. 2. New Search Method.

Figure 1 illustrates the currently deployed search method in unstructured P2P systems. Peer A initiate a query with TTL=3, which means peers within three hops are supposed to receive this query. In Figure 1, link AB, link BC, and link CD are bidirectional. Therefore, D will receive the query with TTL=3 initiated by peer A.

In Figure 2, link BC does not exist. If peer A initiate a query with TTL=3, then peer A may only receive a response from peer B. Currently deployed search method allows peer B to control peer A's access to the rest of the network. Therefore, if peer B refuse to forward peer A's search query to peer C, then peer A will never receive any response from peer C and peer D. If peer A can develop a mechanism that can access peer C and peer D without the help of peer B, then this problem will be solved. In Figure 2, we demonstrate this idea. If peer A and peer C develop a bidirectional link without changing current network topology, then peer B's hidden action will not affect peer A's search performance.

The bidirectional link AC in Figure 2 can be implemented by the following method. For a peer P running MSSF mechanism, it maintains two lists: a Primary Proxy List (PPList) and a Secondary Proxy List (SPList). These lists are composed of IP addresses of other peers that are defined as proxy peers by P itself. Peer P can directly communicate with its proxy peers, because it knows their IP addresses.

Since P2P networks are dynamic with peers joining and leaving frequently, peers should make the proxy peers in PPLists fresh. In MSSF mechanism, a peer P maintains its PPList by periodically selecting a proxy peer Q and sending a Ping_Proxy message to it. If Q does not respond, P will delete Q from its PPList. If Q is alive, it will respond with a Pong_Proxy message that contains a small number of IP addresses selected from its own PPList. P will add these IP addresses to its PPList or SPList with some probability. Hence, peers are able to share proxy peers with each other. If P is not a proxy peer of Q, Q will also add P to its PPList or SPList with some probability. Here the role of the SPList is to temporarily hold a large number of IP addresses of other peers, it is unnecessary and costly to maintain it fresh. Here we use Ping_Proxy and Pong_Proxy messages for distinguishing from Ping and Pong messages that are already used in Gnutella v0.6 protocol [16].

When a new peer joins the system, it may not have any proxy peers. Its first attempt to get some proxy peers is to send Ping_Proxy messages to its neighbors. These neighbors will respond with Pong_Proxy messages, which carry some IP addresses of selected proxy peers. The new peer will add these IP addresses to its PPList and SPList. Of course, the new peer will be introduced to other peers by sending the Ping_Proxy messages to those peers.

When a source peer X submits a query, it initiates a flood with a small TTL and simultaneously sends query messages to some of its proxy peers. When a proxy peer receives the query message from X, it initiates a small-scale flood on behalf of X at once. We let N_{mf} ($N_{mf} \geq 1$) denote the maximum number of floods initiated for one query. Thus, besides the flood initiated by itself, the source peer X requires other $N_{mf}-1$ live proxy peers to initiate floods on behalf of it. In current MSSF mechanism, all floods have the same number of TTL hops (the TTL hops do not include the hop from the source peer to a proxy peer). In addition, we let N_{rd} denote the number of results desired, and S_{ppl} ($S_{ppl} \leq N_{mf}-1$) denote the size of the PPList.

In selecting proxy peers for initiating floods, the source peer should first select the ones from its PPList. Only after all the peers in PPList have been selected can the source peer select the ones from the SPList. The floods are classified into "live" flood and "dead" flood. For the source peer, a "live" flood means sending a query message to a live proxy peer, while a "dead" flood means sending a query message to a proxy peer that is no longer online. If a "dead" flood happens, the source

peer will select another proxy peer to initiate a "live" flood. Hence, there are two conditions that will terminate a query. The first one is that N_{mf} "live" floods have been initiated. The second one is that all the proxy peers have been selected.

Because of the overhead of maintenance, the size of the PPList must be relatively small. Given that there may be many proxy peers, which ones should be inserted into the PPList? In our design, we rank proxy peers based on a method called most query results. The proxy peers that have returned the highest number of results for the previous query will be added into the PPList. Similarly, if a proxy peer in the PPList has returned few results for the previous query, it will be deleted from the PPList.

IV. SIMULATIONS

According to [17], Gnutella-like P2P systems tend to exhibit power-law distribution. Therefore, we use BRITE [18] topology generator to construct an overlay network where the number of links to a peer follows the power-law distribution. To simulate a dynamic network behavior, we let peers die periodically. It is assumed that when a peer dies, another new peer is born and the overlay topology changes. With this scheme, we always keep 25,000 live peers. The average out-degree of these peers is 6. During each run, 2,500 randomly selected peers submit 50,000 queries, and the overlay topology changes about 5,000 times.

We distribute 2,000 different objects of varying popularity in the simulation. A zipfian distribution is used to model both the replication distribution and the query distribution to achieve similar results exhibited in [19]: The most popular 10% of objects amount for 50% of the total number of stored objects and account for over 50% of total queries. With our default parameters, the most popular object is always stored in around 2.4% of the live peers.

In the simulation, we will examine the performance of MSSF with TTL=4. Also, we will examine the performance of Gnutella flooding with TTL=6 as a comparison, which covers about 90% of live peers. The quality of a search mechanism is judged by the following metrics:

Search scope is defined as the number of peers that a query has probed during the search process.

Traffic cost is defined as the number of query messages produced by a query.

Success rate is defined as the ratio of successful queries to total queries. In our simulation, a query for an object is successful if it discovers at least N_{rd} results. In the simulation, the default value of N_{rd} is 10.

Number of hits is defined as the size of the result set for a query.

Optimization rate is defined as gain/penalty ratio, i.e., the ratio of query traffic reduction to overhead traffic increment for maintaining peers' PPLists.

A. Effectiveness of MSSF Without Hidden Action

First we investigate the impact of the maximum number of floods for each query on performance, where the value of N_{mf} changes from 2 to 18. We also need to set value to S_{ppl}. A small PPList means that we can only select a small number of "very good" proxy peers, and a large PPList means low update frequency and high maintaining cost. Therefore, we set S_{ppl} to $N_{mf}/2$. Figures 3 and 4 show the average search scope and traffic cost respectively. In Figure 3 we see that the granularity of search scope is small in MSSF. Increasing N_{mf} will not significantly increase the number of peers visited. This is different from Gnutella flooding. Figure 4 demonstrates that the traffic cost of MSSF is lower than that of Gnutella. In Gnutella flooding, an additional step could exponentially

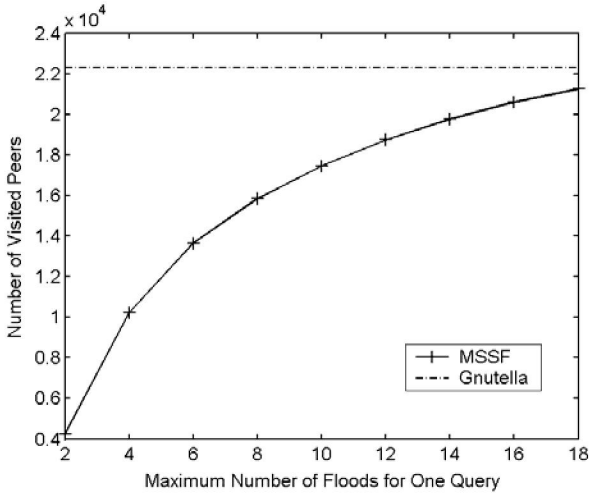

Fig. 3. Search Scope with Different N_{mf}.

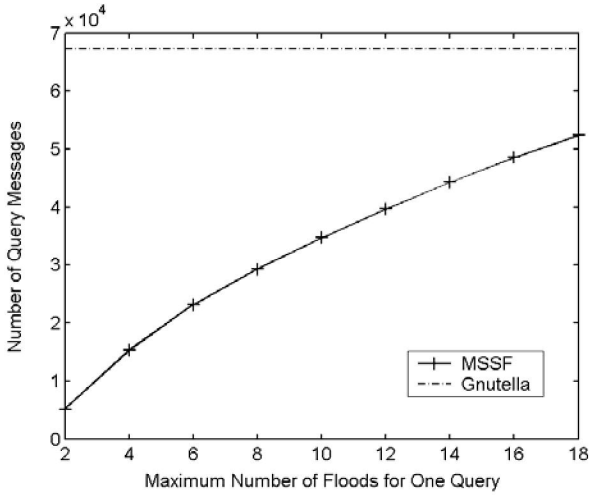

Fig. 4. Query Traffic with Different N_{mf}.

increase the number of query messages; in MSSF, the number of query messages can increase near linearly when increasing the value of N_{mf}.

Figure 5 and Figure 6 show that increasing N_{mf} can bring more results and higher success rate. The sequence and parallel policies return much fewer results than the sync policy and Gnutella flooding. That is because they are adaptive

termination policies that can stop propagating query messages when enough results have been received. However, they do not have obvious reduction on success rate, as shown in Figure 6.

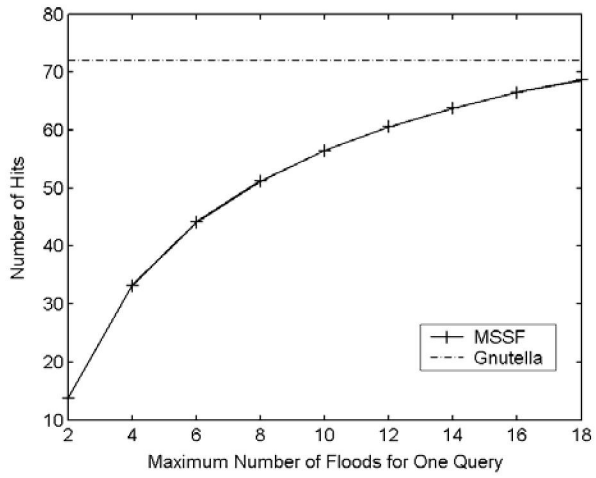

Fig. 5. Query Hits with Different N_{mf}.

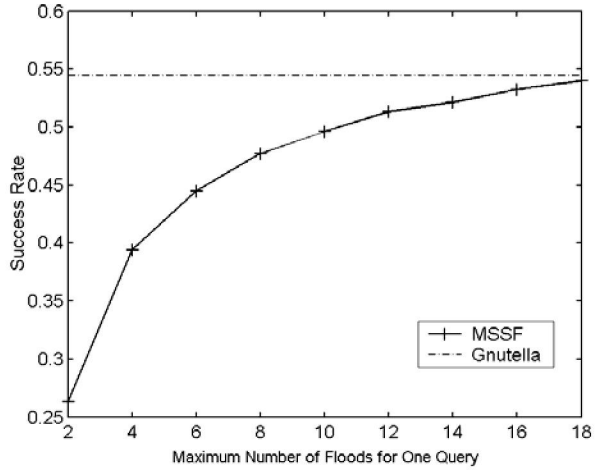

Fig. 6. Success Rate with Different N_{mf}.

When N_{mf} equals 18, MSSF mechanism can receive the comparable success rate (the difference is below 2.5%) of Gnutella flooding, while reducing the traffic cost by 22%

One of the key factors that affect the performance of MSSF mechanism is the extra traffic cost incurred by peers sending *Ping_Proxy* and *Pong_Proxy* messages. In the last set of simulations, we examine the tradeoff between query cost and maintenance overhead of peers' PPLists. We assume that each peer issues 0.3 queries per minute, which is calculated from the observation data shown in [20], i.e., 12,805 unique IP addresses issued 1,146,782 queries in 5 hours. The results in Figure 7 show the usefulness of MSSF. Even checking proxy peers in the PPList 60 times per minute, which is almost the worst case in practice, the optimization rate of the sync, sequence, and parallel policies (N_{mf} =18 and S_{ppl} =9) are still 4.1. Thus, the search improvement achieved by MSSF mechanism will not be outweighed by the maintenance overhead of peers' PPLists.

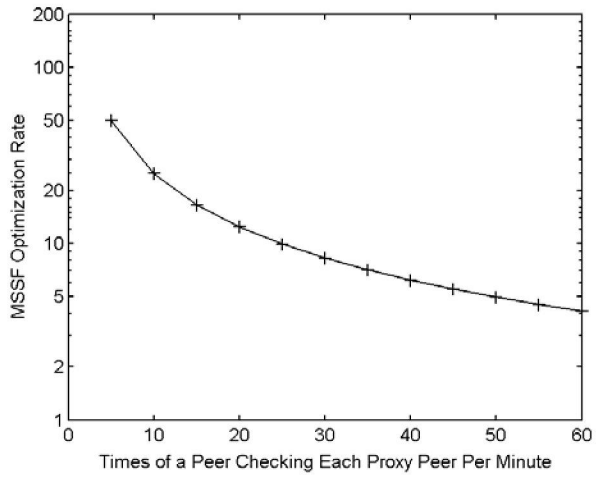

Fig. 7. Optimization Rate.

B. *Effectiveness of MSSF With Hidden Action*

Figure 8 to Figure 10 are experimented where N_{mf} is 18 and S_{ppl} is 9. Figure 8 depicts the search scope degradation with increased peers with hidden action. MSSF outperforms when peers with hidden action go from 10% to 50%. In Figure9, the result of Query Hits is a little different in that MSSF outperforms when peers with hidden action are more than 10%. In Figure 10, MSSF also achieves better performance on

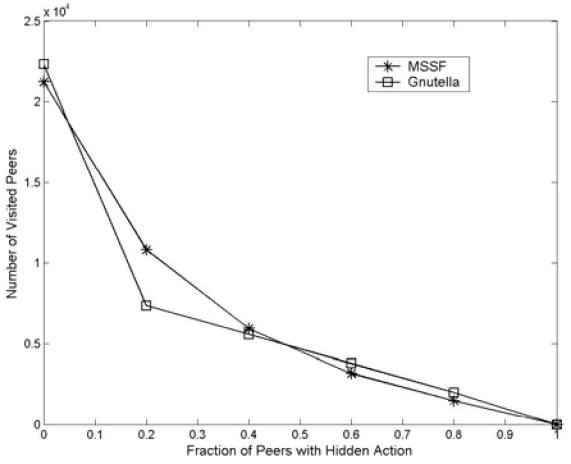

Fig. 8. Search Scope with Hidden Action.

Success Rate.

These three figures illustrate that system performance can be significantly affected by peers with hidden action. When peers with hidden action vary from 0 to 20%, Query Hits and Success Rate degrade fast in Gnutella. MSSF performs much better when peers with hidden action vary from 0 to 40%. However, if peers with hidden action are more than 40%, then MSSF is also ineffective.

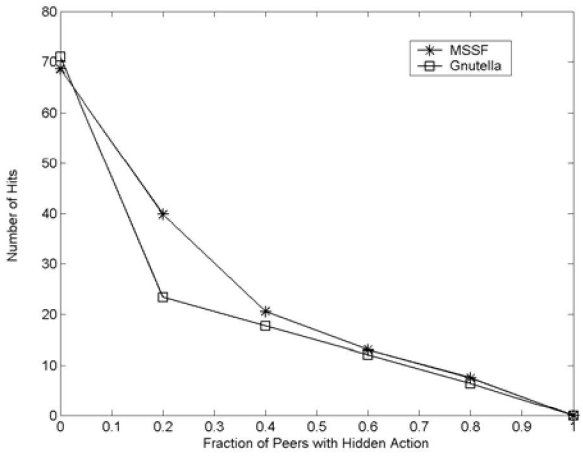

Fig. 9. Query Hits with Hidden Action.

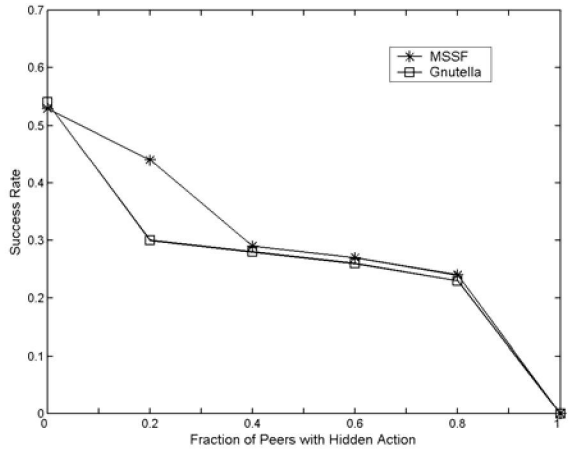

Fig. 10. Success Rate with Hidden Action.

V. CONCLUSIONS

In this work, we identify the problem of hidden-action in unstructured peer-to-peer networks, and show that it is possible to achieve acceptable performance under hidden-action by appropriate search mechanism design. We conclude that besides incentive based methods, improving currently deployed search mechanism is another effective solution to tackle the problem of hidden action.

We are planning to further our work in several aspects. First, we plan to implement some state-of-art incentive mechanisms are to compare these mechanisms with our system and to analyze the advantages and disadvantages of both methods. Second, we plan to give formal and theoretical description of our method. We are planning to build mathematical model and to analyze the strength and weakness of our method using the model. Third, we plan to build a real system and try to find a way to implement our search method and induce incentive mechanism.

REFERENCES

[1] Gnutella, http://gnutella.wego.com/.
[2] KaZaA, http://www.kazaa.com/.
[3] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In Proceedings of MobiCom, 2000.
[4] S. Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes - Fairness in Dynamic Ad Hoc NeTworks. In Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing, June 2002.
[5] S. Buchegger and Jean-Yves Le Boudec. Coping with False Accusations in Misbehavior Reputation Systems for Mobile Ad Hoc Networks. EPFL Technical Report Number IC/2003.
[6] S. Buchegger and J.-Y. Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In Proceedings of WiOpt, 2003.
[7] S. Bansal and M. Baker. Observation-based Cooperation Enforcement in Ad Hoc Networks. Technical report, Stanford University. 2003.
[8] B. Yang and H. Garcia-Molina. PPay: micropayments for peer-to-peer systems. In Proceedings of the 10th ACM Conference on Computer and Communication Security. 2003.
[9] Q. Sun and H. Garcia-Molina. SLIC: A Selfish Linkbased Incentive Mechanism for Unstructured Peer-to-Peer Networks. Technical Report, Stanford University. 2003.
[10] B. Holmstrom. Moral hazard in teams. Bell Journal of Economics, 13(2):324---340, 1982.
[11] R. Strausz. Moral Hazard in Sequential Teams. Departmental Working Paper. Free University of Berlin. 1996.
[12] A. Banerjee and A. Beggs. Simultaneous versus Sequential Move Structures in Principal-Agent Models. University of Oxford, working paper. 1997.
[13] D. Tsoumakos and N. Roussopoulos. A Comparison of Peer-to-Peer Search Methods. In Proceedings of WebDB, 2003.
[14] B. Yang, P. Vinograd, and H. Garcia-Molina. Evaluating guess and non-forwarding peer-to-peer search. In Proceedings of IEEE ICDCS. 2004.
[15] K. Scipanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In Proceedings of IEEE INFOCOM, 2003.
[16] The Gnutella protocol specification 0.6, http://rfcgnutella.sourceforge.net.
[17] M. Ripeanu and I. Foster. Mapping Gnutella Network. IEEE Internet Computing, January/February 2002
[18] BRITE, http://www.cs.bu.edu/brite/.
[19] J. Chu, K. Labonte, and B. Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. In Proceedings of SPIE, 2002.
[20] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. In Proceedings of O'Reilly's Peer-to-Peer and Web Services Conference, 2001.B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.