EPITELLA: Improving the Gnutella Search Algorithm through Epidemic Spreading Models for Complex Networks

Holger Kampffmeyer, Mirco Musolesi and Cecilia Mascolo
Department of Computer Science, University College London
Gower Street, London, WC1E 6BT, United Kingdom
{h.kampffmeyer|m.musolesi|c.mascolo}@cs.ucl.ac.uk

Abstract

Search algorithms in unstructured P2P networks such as Gnutella use flooding-based techniques for communication and, as a consequence, they produce high message overhead. More dynamic algorithms such as Gnutella's Dynamic Query Protocol take into account the user's desired number of results and network topology properties to increase scalability. However, these algorithms only work well for popular files and often fail to locate rare content.

In this paper, we introduce EPITELLA, an enhanced Gnutella which uses a search algorithm based on epidemic-style information dissemination techniques, that shows good performance in finding both rare and popular files. It exploits the structure of the underlying network in order to maximise its search horizon and minimise the number of needed search messages. The presented simulation results show that EPITELLA is more exhaustive in finding rare files and produces less overhead in finding popular files, compared to Gnutella.

1 Motivation and Background

Peer-to-peer (P2P) systems have attracted a great deal of attention in recent years. While what appeals to users is the possibility of unlimited sharing and exchange of files, P2P systems also attract the research community due to the challenges that their decentraliined structure imposes. Much research has been carried out in the area of topology characteristics and measurements, and the development of faster and better search algorithms. P2P networks can be mainly subdivided in two categories: unstructured and structured. In unstructured P2P there is no control over locations of files in the network. Examples of unstructured networks are Gnutella [1], and FastTrack to which Kazaa connects to [5].

The Gnutella P2P network [2], with a user base of over 2.2 million users in March 2006 [5], is one of the top three popular P2P file sharing applications, primarily used to exchange music, movies and software. Locating items in a network is an important part in the operation of P2P networks. When a user wants to search for a file, a QUERY message containing the user specified search string is sent to every node the peer is connected to. A node receiving the QUERY message matches the search string against the locally stored file name and broadcasts the message further. If there is a match, the node answers with a QUERYHIT message. A QUERYHIT message contains information necessary to download the file, such as the number of hits, the IP address of the responding node and the download speed.

Before the introduction of the Gnutella protocol v0.6 in 2002 [2], every node in the network played the role of an equal peer. This resulted in the overloading of peers with slow Internet connection and, generally, in large message overhead, as each node had to forward every message to every neighbour. These scalability problems were improved with the introduction of the *two-tier overlay architecture*.

With the introduction of the two-tier overlay architecture, the Gnutella developer community [1] implemented specialised roles for peers in the network. In this model, a small subset of nodes become *ultrapeers*, responsible for routing all messages and for shielding the *leaf peers* that are connected to them from the network traffic. To become an ultrapeer, a node must provide sufficient bandwidth and uptime and must not be firewalled. Ultrapeers are connected, in average, to 30 other peers [12, 14], while leaf peers hold only a small number of connections ($\leq$ 3) to ultrapeers. Leaf peers, on the other hand, concentrate on providing files. Information about the files the nodes share are uploaded to the ultrapeer. An ultrapeer forwards only those QUERIES to its leaf pears that have matching files. The introduction of the two-tier topology was the first step to improve the scalability and performance of Gnutella by reducing search message overhead. In order to improve the Gnutella performance, the *Dynamic Query Protocol* (DQP) was introduced to limit the flooding of query messages. The Dynamic Query Protocol reduces message overhead by

considering the popularity of the queried file and by lowering the used message TTL.

The goal of this protocol is to only gather enough results to satisfy the user's needs (typically 50 to 200 results). The DQP delegates all search responsibilities to the ultrapeers. Every leaf peer commits its file list to the ultrapeer it is connected to. When an ultrapeer receives a search query from an attached leaf node, it starts a dynamic query which consists of two steps: first it sends a query message with a low TTL that is flooded outwards, decreasing the TTL each time it is forwarded until the TTL expires. This is called a *probe query*, meant to give an idea of the popularity of the file. Depending on the results the ultrapeer receives, a second query is started with a higher TTL. This process is repeated until the desired number of results is reached or the ultrapeer gives up. However, the DQP may have poor performance especially in case of rare files in terms of message overhead.

In this paper we propose EPITELLA, a decentralised, unstructured search algorithm, which uses epidemic-style information dissemination techniques to control the number of query messages and to fine-tune the dissemination process. The approach is based on recent results of complex networks theory and models of epidemics spreading. We present simulation results that show that EPITELLA is more exhaustive in finding rare files and produces less overhead in finding popular files, compared to the Dynamic Query Protocol currently used in Gnutella.

The reminder of this paper is organised as follows. We describe the EPITELLA search algorithm in Section 2. The evaluation in Section 3 shows that our algorithm produces better search results for rare files, and produces less overhead in finding popular files compared to Gnutella. Section 4 concludes the paper, summarising its contribution.

2 EPITELLA

Epidemic algorithms represent an effective solution to disseminate information in distributed systems[9] and we argue they are also suitable for P2P networks. In fact, there is a strong analogy between information dissemination in distributed systems and epidemic transmission in communities. Diseases are transmitted by contacts between infected individuals and susceptible (i.e, that can be potentially infected) individuals. The spread of an epidemic can also be used to model replication and dissemination of information in a P2P system. The highly resilient nature of epidemics can be used to ensure the reliability and robustness of an epidemic algorithm. In a community, only a few infected individuals are sufficient to spread the disease to a large number of members in a very short time. The analogy becomes even more evident when the content of the information is malicious as in the case of computer viruses [8].

Epidemiologists have found that there is a critical threshold for the propagation of a disease throughout a population dependent on the infectivity of the disease [6]. Any epidemic less infectious than this threshold will inevitably die out, whereas those above the threshold will increase exponentially. Recent study in complex network theory have also linked this threshold to the structure of the network [7]. For example, in scale-free networks this threshold is zero. That is, all viruses, even those that are weakly contagious, will spread and persist in the system.

2.1 The Infection Spreading Model

The epidemic dissemination algorithm that we adopt as a basis for a search algorithm in a Gnutella-like overlay network, is inspired by the work presented in [11], which was applied to mobile ad hoc networks. The core of the epidemic dissemination algorithm is the Susceptible-Infective-Removed (SIR) infection spreading model, proposed by Kermack and McKendrick in 1927. A simplified version is the Susceptible-Infective-Susceptible *SIS* model [6]: according to it, an individual can be in two possible states: infected (i.e., an individual is infected with a disease), and susceptible (i.e., an individual can potentially get infected). In this simplified model, we do not consider the case of nodes that cannot be infected again after recovering from the disease (i.e., the recovered state) or because they disappeared from the system (i.e., they are isolated or died).

We map the model onto P2P networks, by substituting the population of individuals with the nodes of the network. A host is considered infected, if it holds a message, and susceptible, if it does not. If the message is deleted from the host, the host becomes susceptible again. The dynamics of the infectives and susceptibles in a scenario composed of $N(t)$ active hosts (i.e., not failed), can be described by means of a system of differential equations:

$$\begin{cases} \frac{dS(t)}{dt} = -\beta S(t)I(t) + \gamma(t)I(t) \\ \frac{dI(t)}{dt} = \beta S(t)I(t) - \gamma(t)I(t) \\ \frac{dN(t)}{dt} = -\phi N(t) \\ S(t) + I(t) = N(t) \end{cases} \quad (1)$$

where:

- $I(t)$ is the number of infected hosts at time t;
- $S(t)$ is the number of susceptible hosts at time t;
- β is the average number of contacts with susceptible hosts that leads to a new infected host per unit of time per infective in the population;
- γ is the average rate of removal of infectives from circulation per unit of time per infectives in the population;

- ϕ is the failure rate (i.e., the probability that one host fails per unit of time).

If we solve the system by using the initial condition $I(t) = I_0$ (where I_0 is the number of initial hosts infected), we obtain that the number of infectives at time t is described by the following equation:

$$I(t) = \frac{I_0 e^{\alpha\beta t}}{1 + \frac{I_0}{\alpha}(e^{\alpha\beta t} - 1)} \quad (2)$$

with $\alpha = N(t) - \frac{\gamma}{\beta}$. $N(t)$ is considered approximately constant during the entire epidemic process described by the system 1, since we assume that the failure process is stationary considering the interval of time during which the epidemics spreading happens (i.e., we assume $N(t) \approx N^*$ with N^* equal to the number of hosts present in the system at the beginning of the epidemics). In our case the initial condition is $I_0 = 1$: this represents the first copy of the message that is inserted in its buffer by the sender. This result can be used to calculate the number of infectives at instant t with a given infectivity β and a given removal rate γ, or, more interestingly for our purposes, β and γ can be tuned in order to obtain a certain epidemics spreading, after a specific length of time has passed. The infectivity β is the fundamental parameter of the message replication algorithm. In fact, a certain infectivity β can be selected in order to obtain, at time t^*, a number of infectives (i.e., hosts that have received the message) equal to $I(t^*)$ or, in other words, a percentage of infectives[1] equal to $I(t^*)/N(t^*)$. The parameter γ can be interpreted as the deletion rate of the messages from the buffer of the hosts. We assume sufficient dimensioned buffers so we do not need to delete messages from the buffer during the dissemination process. We therefore assume $\gamma = 0$.

In homogeneous networks, such as random graphs[2], the node degree k for each node can be approximated quite precisely with the average degree of connectivity $\langle k \rangle$ of the network. Therefore, in case of homogeneous networks, in order to take into account the effect of the connectivity, it is possible to substitute β with $\lambda\frac{\langle k \rangle}{N}$. λ represents the probability of infecting a neighbouring host. $\frac{\langle k \rangle}{N}$ gives the probability of being in contact with a certain host. Thus, we have separated, in a sense, the event of being connected to a certain host and the infective process [7], and it is therefore possible to calculate λ as function of $I(t^*)$ and $\langle k \rangle$. Finally, it is interesting to note that in homogeneous networks, every host knows its value of k and, consequently, of $\langle k \rangle$. We will exploit this property to tune the spreading of message replicas in the system.

[1]Note that $\beta = f(I(t))$ is not defined for $I(t) = N(t)$. Therefore, from a practical point of view, in the case of a message sent to all the hosts of the system, we will use the approximation $I(t) = N(t) - \epsilon$, with $\epsilon > 0$, in the expression used to calculate β.

[2]The degree distribution of a random graph is a binomial distribution with a peak at $P(\langle k \rangle)$.

2.2 EPITELLA Search Algorithm

We now describe the *EPITELLA search algorithm*; the algorithm takes into account the desired number of search results and the popularity of the queried file and tunes the dissemination of the query messages accordingly. We exploit a two-step search process inspired by the Gnutella's Dynamic Query Protocol, to dynamically adapt our algorithm to different file popularities.

In a P2P network such as Gnutella, messages are sent from a source A to its neighbours in a flooding-based manner. To avoid the overhead produced by flooding the message to each neighbour, we use the results described in Section 2.1 to only deliver a message from A to a percentage of nodes Ψ, by time t^*. In fact, more formally, given an expected percentage of hosts which has to be infected equal to Ψ, the value of β in order to obtain $I(t^*) = \Psi N$ can be calculated.

Firstly, a *probe search* limited to a small number of hosts is triggered in order to get an estimation of the file popularity. A second query is designed, by taking into account the number of responses from the probe search, to generate just enough results to satisfy the users needs (typically 50 to 150 number of results). In case of popular files, where the probe query already delivers a sufficient number of results, no second query is sent and the search process stops immediately.

However, there is a major difference between the Delivery Query Protocol used in Gnutella and our search approach. Instead of using a higher or lower TTL value to adapt to rare or popular files, we use the parameter β of the epidemic model to fine-tune the dissemination of the query. The EPITELLA algorithm also takes into consideration other network parameters such as the number of nodes N and their connectivity k. In case of a poorly connected overlay the algorithm is able to adapt the dissemination process to still make sure a sufficient number of nodes receive the query.

Recent studies [15] have found that files in Gnutella are randomly distributed in the network. Therefore, a simple proportion is used to evaluate the probe query and to choose a sufficient value of the reliability parameter for the second query:

$$\frac{R_{probe}}{R_{desired}} = \frac{\Psi_{probe}}{\Psi_{desired}} \quad (3)$$

Hence:

$$\Psi_{desired} = \frac{\Psi_{probe} R_{desired}}{R_{probe}} \quad (4)$$

R_{probe} is the number of results that are returned from the probe query, and $R_{desired}$ is the number of results the user wants to receive. Ψ_{probe} is the value of the Ψ parameter for the probe query (i.e., the percentage of hosts that are reached by the probe query). $\Psi_{desired}$ is the percentage of hosts that need to be reached in order to have the desired number of results.

For example, let us consider a network composed of 10000 nodes and let us suppose that we would like to retrieve 100 results. Given the number of hosts in the network and the average link distribution, we calculate the value of β that is necessary and sufficient to reach a given small number of nodes in the probe phase (for example 500, equal to 5% of the total number of hosts). Then, let us suppose that the probe search returns 50 results. In order to obtain the desired 100 results, using the simple formula above, we need to reach 10% of the nodes of the system. Then, we calculate the infectivity needed to reach this percentage of hosts and we spread the message using this infectivity.

This simple solution provides the best results if the files are perfectly uniformly distributed in the network, since it is able to retrieve the desired number of results with the minimal message overhead. The accuracy of the protocol is also a function of the size of the probe sample and the popularity of the file for statistical reasons. The choice of the best values of Ψ_{probe} can be performed experimentally. It is also possible to use different values of Ψ_{probe} for different popularity of files. In fact, if the files is highly popular, a low value of Ψ_{probe} is able to provide accurate results. On the flip side, it is necessary to use higher values of Ψ_{probe} for files with a low popularity. The popularity of a file can be set by the user or can be retrieved automatically by the system. For example, a possible method for estimating file popularity is presented in [10]. In general, as the size of the population of the probe sample and the number of nodes increases, the accuracy of the protocol will increases. We will discuss some simulation results about all these issues in Section 3.

Each time a node wants to send a new message over the network, the system must calculate the infectivity λ which is necessary and sufficient to spread the information with the desired reliability (that has to be chosen in the range [0,1]), in a specified time interval, evaluating the average degree of connectivity and the number of nodes of the network. The time interval is measured in logical time unit (or rounds). The calculation of the infectivity using the epidemic model is also based on logical time units. One logical time unit corresponds to a physical time unit (for example 0.01 seconds). At every round, the dissemination process is executed.

The message's unique identifier, the value of the calculated infectivity, the expiration time that indicates when the message needs to get deleted from the network (i.e., at the end of the search after the total number of round of the epidemic spreading) are inserted into the header of the message and the message is stored in the node which creates the message.

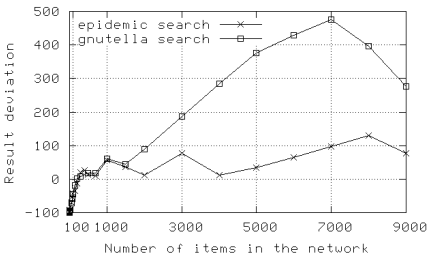
To implement EPITELLA, we adapt the Dynamic Query Protocol implementation of PHEX [4], a Gnutella client written in Java, into PeerSim [3]. PeerSim is a Java framework designed to experiment with large scale P2P overlay networks. Its simulator architecture is highly scalable (i.e., it is able to support a high number of network nodes), and, through its component based structure, fully configurable. This makes it easy to join together different pluggable building blocks. For Gnutella, we use the PHEX message implementation. Instead of using a TTL field as in the Gnutella message implementation to control the validity of the message, we can use the expiration time and delete the epidemic message at time t^*.

3 Evaluation

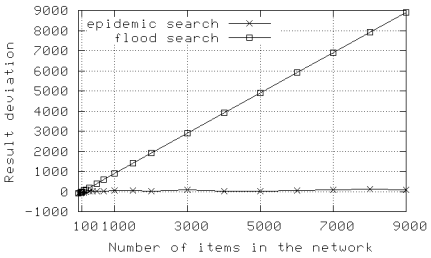
In order to evaluate the epidemic algorithm we present some preliminary simulation results gathered using PeerSim. We show that the Epidemic Algorithm can be applied as an effective search algorithm which can compete against Gnutella's search engine. We compare the Epidemic Algorithm as a search algorithm in a Gnutella-like network with Gnutella's Dynamic Query Protocol. For a reference, we also show a comparison between the EPITELLA Search and Flood Search, a flooding-based algorithm that was used in the early Gnutella implementations.

We choose a Random Graph network as the basic overlay network for our experiments with Gnutella. In fact, as described in Section 1, only ultrapeers are responsible for searching the Gnutella network. Therefore, we only consider the ultrapeer overlay in our simulations. In [14, 13], Stutzbach and alii have found that this ultrapeer-layer forms a stable core-layer, whose nodes are connected randomly. This top-level overlay forms a stable core, with a spike in its connectivity degree distribution around 30. This means, we can approximate the Gnutella ultrapeers topology through a Random Graph with an average degree of connectivity $\langle k \rangle = 30$. To compare *EPITELLA Search* with the Gnutella Search, we consider three parameters which are fundamental for an efficient, reliable and slim search algorithm:

- *Deviation from desired to received results*: This is the error of the number of results the user wants with respect to the actually number of received results. The lower this deviation is, the more effective is the search.
- *Sent messages*: This is the number of overall sent messages in the network. The less messages a query needs to successfully return the number of desired results, the more efficient an algorithm is.
- *Sent/Received ratio*: This is the number of sent mes-

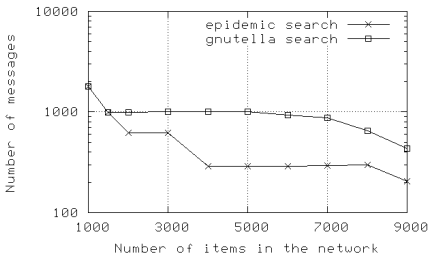


(a) Gnutella vs. EPITELLA Search

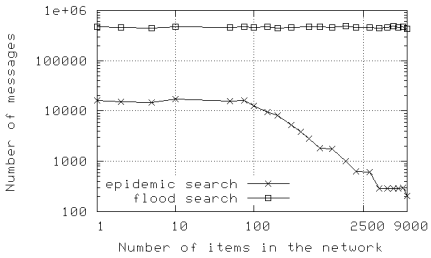


(b) Flood Search vs. EPITELLA Search

Figure 1. Comparison of the deviation from desired results (100) in Gnutella, Flood Search and EPITELLA Search, with a variable number of files in the network.



(a) Gnutella vs. EPITELLA Search

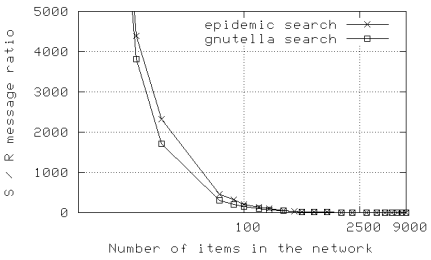


(b) Flood Search vs. EPITELLA Search

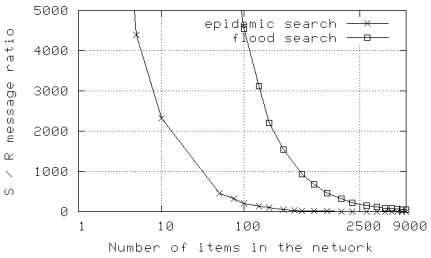
Figure 2. Comparison of the sent messages in Gnutella, Flood Search and EPITELLA Search, with a variable number of files in the network.

sages, divided by the number of received results. A low sent / received ratio means a highly efficient query. Each of these parameters are evaluated in a network with 10000 nodes, using an average degree of connectivity $\langle k \rangle = 30$, and a time period of $t^* = 100$ rounds. Each of the parameters is printed as a function of *number of items* in the network that is the file popularity. Apparently, the overhead produced by a high deviation of desired results and the resulting number of sent messages is far higher for popular files. On the other hand, a low sent/received ratio is only meaningful in combination with a low deviation of desired results. For a user, it is more important to receive the desired number of results, so it is acceptable for a query to need more query messages for rare files. The potential in saving messages lies in finding popular files efficiently, without producing high overhead.

The system counts the overall number of sent messages, the number of query hit messages and calculates the deviation from the desired number of results. Each simulation for each sample consists of 50 experiments, each experiment takes 200 rounds (100 rounds for the probe and the standard search phase, respectively). For the Flood Search, a TTL-constrained search algorithm similar to the one used in early Gnutella implementations, we use a $TTL = 7$, which is the maximum value the TTL field is allowed to have [2]. Figure 1.a shows the deviation from the number of desired results a user wants (in our case 100 results). With increasing file popularity the EPITELLA Search is far closer to the number of desired results than Gnutella, meaning that Gnutella still produces a considerable overhead when searching for popular files. The efficiency of EPITELLA Search becomes more apparent when compared to Flood Search in Figure 1.b: while the deviation of desired results appears to stay almost constant in EPITELLA Search, Flood Search produces a linear increase of the result deviation. This clearly shows the advantage of the dynamic two-step process of EPITELLA Search and DQP. For popular files, the flooding-based algorithm produces far more results a user is ever likely to use. Figure 2.a shows the overall number of sent messages of EPITELLA Search and Gnutella using the same simulation settings as in Figure 1. For popular items again, EPITELLA Search shows its supremacy. It needs significantly less messages than Gnutella. The particular behaviour of Gnutella is caused by the use of a TTL-based search. The number of obtained results is higher than the desired ones. In this case the search is extended to a higher distance (i.e. TTL), but this results not only in a higher number of results but also in a higher associated overhead. Then, after 7000, with a lower TTL, Gnutella is able to retrieve the desired number of results with a more limited exploration. This reduces the deviation from the number of desired results and the overhead. The



(a) Gnutella vs. EPITELLA Search



(b) Flood Search vs. EPITELLA Search

Figure 3. The sent/received message ratio in Gnutella, Flood Search and EPITELLA Search, with a variable number of files (items to search) in the network.

magnitude of savings of messages is even more impressive in Figure 2.b, when compared to the sent messages of Flood Search. Figure 3.a shows the sent/received ratio of Gnutella and EPITELLA Search. The performance of both of the algorithms are very similar for popular files. EPITELLA Search exhibits a slightly higher sent/received ratio for rare files. This is a wanted behavior, because the user is interested in actually retrieving search results, if there are files in the network that match the user's query. It is therefore acceptable to send more search messages than Gnutella sends, if the results are better and more search results are produced. In these experimentation, we used different values of Ψ_{probe} for different values of file popularity, with 1% for low, 7% for medium and 15% for high popularity. However, it may happen that the popularity of a file cannot be retrieved with a good accuracy.

The results show that the EPITELLA search algorithm performs well and is able to produce better results than even the highly optimised Dynamic Query algorithm of Gnutella. While the Gnutella search algorithm is optimised for the specific topology characteristic of the current Gnutella network, our EPITELLA is not restricted to any specific network characteristic. It works best in a highly connected Random Graph network (for statistical reasons), but due to its dynamic adaptation mechanisms, it is also able to handle less connected graphs.

4 Conclusions

In this work we have introduced a search algorithm based on epidemic-style information dissemination [11]. Being inspired by the two-step process of Gnutella's Dynamic Query Protocol we have designed a search algorithm that relies on a probabilistic dissemination techniques based on epidemiological models in complex networks. We have implemented this algorithm using PeerSim, a Java-based P2P simulator. Finally, we have compared this solution with Gnutella's approach and with a flooding based search algorithm. As an evaluation, simulation results have been given for all three search algorithms.

References

[1] Gnutella Development Forum, March 2006. http://www.the-gdf.org/wiki/.

[2] Gnutella Dynamic Query Protocol v.0.6, March 2006. http://www.the-gdf.org.

[3] PeerSim Project Webpage, March 2006. http://peersim.sourceforge.net/.

[4] PHEX Gnutella Client, March 2006. http://phex.kouk.de.

[5] Slyck - File Sharing News and Info, March 2006. http://www.slyck.com.

[6] R. M. Anderson and R. M. May. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, 1992.

[7] M. Barthélemy, A. Barrat, R. Pastor-Satorras, and A. Vespignani. Dynamic Patterns of Epidemic Outbreaks in Complex Heterogeneous Networks. *Journal of Theoretical Biology*, 2005.

[8] T. M. Chen and J.-M. Robert. Worm Epidemics in High-Speed Networks. *IEEE Computer*, pages 48–53, June 2004.

[9] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massouli. Epidemic Information Dissemination in Distributed Systems. *IEEE Computer*, May 2004.

[10] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing P2P File-Sharing with an Internet-Scale Query Processor. In *Proceedings of VLDB'04*, pages 432–443, 2004.

[11] M. Musolesi and C. Mascolo. Controlled Epidemic-style Dissemination Middleware for Mobile Ad Hoc Networks. In *Proceedings of MOBIQUITOUS'06*. ACM Press, July 2006.

[12] D. Stutzbach and R. Rejaie. Characterizing the Two-Tier Gnutella Topology. June 2005.

[13] D. Stutzbach, R. Rejaie, and A. H. Rasti. On the long-term evolution of the Gnutella network. In *Proceedings of 8th IEEE Global Internet Symposium*, April 2006.

[14] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Proceedings of IMC'05*, October 2005.

[15] S. Zhao, D. Stutzbach, and R. Rejaie. Characterizing files in the modern Gnutella network: A measurement study. In *SPIE/ACM Multimedia Computing and Networking*, January 2006.