Using Random Walks to Find Resources in Unstructured Self-Organized P2P Networks*

Vicent Cholvi
Universitat Jaume I
12071, Castellón, Spain
Email: vcholvi@lsi.uji.es

Antonio Fernández, Luis López, Luis Rodero-Merino
LADyR, Universidad Rey Juan Carlos
28933, Móstoles, Spain
Email: {anto,llopez,lrodero}@gsyc.es

Abstract

In this paper we present on-going work on the use of random walks as the basic mechanism to locate resources in Peer-to-Peer (P2P) systems. In this work we combine random walks with a self-organization of the overlay network, which dynamically adapts the topology of the network to the load of the system. We present the self-organization techniques we have proposed, and show by empirical evaluation that in fact they lead to topologies in which random walks perform very efficiently. We conclude by describing our current and future lines of research to complete this work.

1 Introduction

Peer-to-Peer (P2P) systems [2] are a new paradigm in which any member (peer or node) of the system can both offer and request resources (such as files, services, etc.). This contrasts with the traditional client-server approach, in which each participant has a well-defined (either server or client) role. P2P systems are usually implemented without centralized control, which leads to interesting properties like flexibility, scalability and fault-tolerance. The nice properties of P2P systems, and the growing impact of P2P-based systems on the Internet (*e.g.* for massive file-sharing), has driven the research community to invest an important amount of effort in attempts to improve the efficiency of P2P systems.

One of the fundamental problems in a P2P system is the efficient location of the resources requested by peers. P2P systems can be mainly grouped into two classes by considering the techniques used for resource location:

- *Structured Systems*. These systems assign responsability of each resource to specific peers. Then, directed routing algorithms allow to locate a given resource by directing the search to some peer responsible of the resource.
- *Unstructured Systems*. These systems do not have strict control on where (the information about) a resource is placed. Then, the search of a resource cannot be directed to the appropriate peer, and mechanisms like flooding or random walks have to be used to locate it.

Most structured systems are very efficient, since they are able to find a resource in only $O(\log n)$ steps (where n is the number of peers in the network). Additionally, they do not suffer of *false negatives* (*i.e.,* a resource present in the system is always found). Unstructured systems, on the other hand, usually adapt better to situations of high churn[1], and allow to perform searches by keywords or regular expressions (instead of exact matching) in a simpler way than structured systems. A description and comparison of several P2P systems of both classes can be found in [2].

Most unstructured P2P systems locate resources by *flooding* part of the underlying overlay network. In the simplest case, the location algorithm simply floods the network up to a certain distance from the peer that started the search. This leads to scalability problems (the network quickly saturates as the number of searches increase) and may suffer of false negatives (if the desired resource is beyond the search range). An improvement of this technique is to select a set of *super-peers* among the peers (ideally, the peers with higher computational and communication capacities). Then, every peer provides at least one super-peer with information about the resources it offers, and resource location is performed only among the super-peers, typically via flooding. This approach reduces the scalability problems, but increases the vulnerability of the system to targeted attacks.

An alternative technique for resource location in unstructured P2P systems is the use of *random walks.* A random walk is a path in a network such that at each point in the path the next link has been chosen uniformly at ran-

*This work was partially supported by the Comunidad de Madrid under grant S-0505/TIC/0285 and the Spanish MEC under grants TSI2004-02940, TIN2005-09198-C02-01, and TSI2006-07799.

[1] Rate at which peers enter and leave the system.

dom from the outgoing edges of the current node. With this technique, a peer that wants to find a resource sends a search message that traverses a random walk in the overlay network, until either the resource is found or the search is dropped. Typically, a peer informs all its neighbors in the network about the resources it holds[2]. Hence the resource is found as soon as the random walk reaches a neighbor of any peer holding it. Random walks introduce less load into the network than flooding, but in general they need longer paths (and hence more time) to find a resource. Additionally, since it is difficult to predict in advance how long a random walk can take to find a resource, there is no way to know if a failed search is a false negative.

However, some studies [5, 8] have shown that random walks are a promising technique for resource location, providing scalability. It is also known that the performance of this technique strongly depends on the topology of the overlay network [1, 7]. This has led us to start lines of work for devising effective ways to improve the performance of random walks. One of these lines, for instance, attempts to improve the performance by adapting the topology of the overlay network to the load of the system.

Related Work. There are other works that have proposed to combine dynamic overlay topologies with random walks for resource location. For instance, Lv *et al.* [9] have proposed a distributed flow-control and topology-maintenance algorithm, by which every node monitorizes the traffic exchanged with each of its neighbors. When some peer i receives too many search messages from some neighbor j, it tells j to reconnect their common link to some other neighbor k of i, which is chosen considering the spare capacities of all neighbors of peer i.

A second relevant proposal is the system *Gia*, by Chawathe *et al.* [3], which is a refinement of [9]. In Gia, every peer periodically sends some amount of *tokens* to each of its neighbors. A token represents a search message that the corresponding neighbor can forward to the peer. The more capacity some peer has, the more tokens it will send to its neighbors. The distribution of tokens is not necessarily even among neighbors, and depends on the neighbor's capacity: the larger the neighbor's capacity, the more tokens it will receive. Peers implement a flow-control mechanism, so if they start to receive too many search messages, they reduce the token sending rate. At the same time, all peers continuously run an adaptation algorithm, which is used to converge to states in which peers send the same number of tokens they receive.

It is interesting to note that in the two above proposals, search messages do not follow *pure* random walks. Instead, at each hop a search message is resent to the neighbor of greatest degree. This decision is inspired by the work of Adamic *et al.* [1], and follows the intuition that, if a peer has greater degree, it also has a better knowledge of the resources in the system, and hence will be able to finish a search with higher probability.

The line of research presented in this paper builds on the work of Guimerá *et al.* [7], which characterizes the network topologies that minimize the average time needed to complete a search if messages follow a shortest path. They assumed a homogeneous system[3] and found out that, when the system is not congested, the optimal topology is a star-like structure (since central nodes know about all the resources, all searches are completed in only one hop). On the other end, when the load grows, the optimal topology is a random-like one. Moreover, they showed by simulation that the transition is very sharp. Their results are not constructive, since they do not describe how these topologies can be obtained.

Results. Keeping the results of [7] in mind, our initial attempts to improve the performance of random walks propose techniques to adapt the topology of the overlay network. These techniques make the network evolve to a centralized star-like topology when the P2P system is underloaded, and evolve to a random-like topology as the load increases. Unlike Gia, the techniques proposed will not require that nodes maintain updated information of their neighbors' state, and can in fact lead to optimal centralized topologies in lightly loaded systems.

Our first contribution has been a reconnection mechanism that is able to achieve these goals without central coordination. Unfortunately, the mechanism assumes knowledge of the load in every node of the network. Then, we have proposed a second algorithm which is able to converge to the desired topologies in the extreme cases while removing the total knowledge requirement. This is achieved replacing the total knowledge by random samples obtained with random walks. Finally, a third algorithm has been proposed that is, additionally, able to properly deal with dynamic heterogeneous systems in which peers may have very different processing capacities and upload bandwidth. This algorithm has the desirable feature that it is very robust in the presence of directed attacks.

Structure. The structure of the rest of the paper is as follows. In the next section we present the technique that allows to adapt the topology if total knowledge of load levels is available. Then, in Section 3 the algorithm that removes this requirement is presented. Section 4 presents the algorithm that properly deals with heterogeneous systems. Finally, in Section 5 future lines of work are presented.

2 Removing the Central Coordinator

As we said, Guimerá *et al.* showed that a centralized overlay topology is the most appropriate when a system is

[2]This is usually termed *one-hop replication.*

[3]All nodes have similar processing capacity.

lightly loaded and that a random-like is more convenient when it is congested. However, applying these results to real P2P systems is not direct. There are two main obstacles to it. First, in their model search messages traverse a shortest path to the peer holding the resource. This is clearly not possible in an unstructured P2P systems, in which peers have little knowledge of the overlay topology beyond their own neighborhood. Second, and more importantly, without centralized control it is not clear how to identify when a P2P system switches between loaded and underloaded states, and how to change its topology accordingly.

In [4] we have partially removed the second obstacle, by proposing a *reconnection mechanism* that is able to adapt the topology without centralized control. In our system, the overlay network is formed as follows. Each node handles a set of *native* connections. When some node i links one of its native connections to some other node j, that connection becomes a *foreign* connection for j. Thus, each node can have both native and foreign connections, each one linking it with other peers of the network. The proposed reconnection mechanism describes how a nodes can change its native connections. To do so, it takes local decisions that depend on the load of every peer in the system. From these local decisions a global behavior emerges, that drives the network to the intended topologies. Thanks to this reconnection mechanism, there is not need of a central coordinator for the topology change.

These are the details of the reconnection mechanism. We assume that the load ℓ_i of every peer i is known and that there is also a known threshold τ that can be used to discriminate whether a node is congested (*i.e.,* node i is congested iff $\ell_i \geq \tau$). Every peer i is assigned either a weight of $w_i = 1$ if it is congested or a weight of $w_i = \delta_i^2$ if it is not, where δ_i is the current degree of the peer. Finally, every node i is assigned a probability p_i proportional to its weight, $p_i = w_i / \sum_j w_j$. Then, a peer j chooses the peers to connect its native connections randomly with probabilities p_i.

In a real P2P system this mechanism would be executed periodically. It is shown in [4] that in the extreme cases in which either none or all nodes are congested, the overlay network quickly converges to the desired corresponding topology. See [4] for details.

Note that this reconnection mechanism still assumes global knowledge, since all nodes know the load of the rest. Additionally, the evaluation of the mechanism done in [4] still assumes that messages follow shortest paths. Thus, we still have to modify the reconnection mechanism proposed in [4] to make it work when nodes only have local knowledge and searches follow random walks.

3 Sampling with Random Walks

To remove the need of knowing the load of each peer in order to use the reconnection algorithm, we have proposed the following simple technique [10]. When the reconnection mechanism is run, peer i chooses its new native neighbors only from a set of candidates C_i, instead of considering all the peers in the system. This set represents a random sample of all the peers in the system. Then, the probability of i choosing j is $p_{i,j} = w_j / \sum_{k \in C_i} w_k$ if $j \in C_i$ and $p_{i,j} = 0$ otherwise.

Each peer collects its set of candidates by sending a special sampling message with a bounded TTL (maximum number of hops) to traverse the network, following a random walk. When the TTL expires, the list of nodes traversed and their load is sent back to the source peer. The peers in that list become the set of candidates. This way of collecting candidates has two features: (1) it demands few resources, and (2) with high probability nodes of high degree (potentially the most interesting as candidates, as they know much about the network) will be part of the set of candidates, as it its very likely that they will be visited by the random walk. The results of Gkantsidis *et al.* [6] show that sampling with random walks provides a high degree of randomness to the sample.

To evaluate this and the previous techniques in a real system, we implemented a first version of a system that we call DANTE[4]. DANTE is a P2P system whose peers run the reconnection mechanism proposed. This mechanism is triggered periodically at each node, and decides to which other peers that node must connect. In DANTE the load of a node is measured as the number of search messages received by the node during some recent period of time. If the load is greater than the threshold τ, the node is said to be congested. As proposed, the system performs peer sampling by using special search messages that follow a pure random walk, bounded by a TTL mechanism. DANTE also uses search messages following pure random walks for resource location. The implementation of DANTE has confirmed the applicability of the techniques proposed in a real network, but also allowed us to detect some limitations of the proposal.

To test the performance of this system, several experiments were run in a small cluster of 7 PCs. Each experiment consisted on a small network of 42 peers (6 peers per PC). The system was homogenous, since all peers were identical. Several values for the threshold τ were tested: 0, 10, 50, 100, 1000 and 10000. As expected, threshold $\tau = 0$ forced the network to form a random topology, and the threshold $\tau = 10000$ forced the network to form a star-like topology. Each threshold was tested under different loads. The loads were due to the nodes, which sent a certain amount of search messages (queries) per unit of time. These amounts were of 2, 4, 6, 8, 10 and 12 queries per minute. Each query looked for a uniformly chosen resource held at only one node (no replication).

Some of the results obtained are shown in Fig. 1, where it can be compared how the performance for each threshold

[4]From Dynamic Adaptable Network TopologiEs

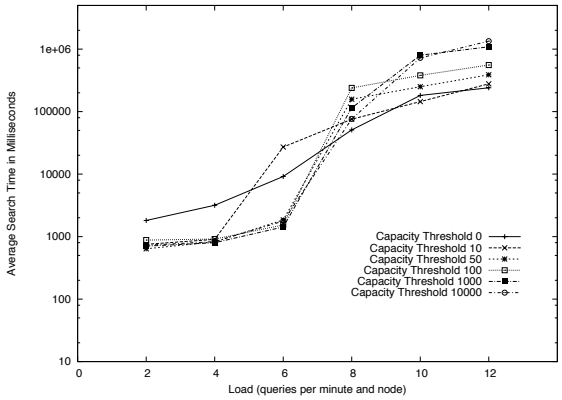

Figure 1. Average time to complete a search in DANTE.

evolved as the load was increased. As expected, the centralized networks ($\tau = 10000$) perform better than the random one ($\tau = 0$) for low loads. Yet, for high loads, the centralized topology has the worst performance, while the random topology gives the lowest search times. Besides, we see that for middle thresholds the topology is adapted to the load, so they tend to build centralized topologies for low loads and random topologies for high loads.

While the results obtained seemed promising, they were obtained in a very small system of homogeneous peers with no churn. It was still an unsolved problem how to deal with size, churn and heterogeneity. Additionally, the experiments showed that it was difficult to find the right value of the threshold τ for optimal performance.

4 Dealing with Heterogeneity

In an attempt to deal with some of the deficiencies of the first version of DANTE, a second version of the system has been proposed [11]. The main difference in this version is the way the weights assigned to the different peers are computed. As before, a sample of peers C_i is obtained by each peer i using a random walk. However, the weight of a peer $j \in C_i$ is computed as

$$w_{i,j} = \delta_j^{\gamma_{i,j}},$$

where $\gamma_{i,j}$ is computed as

$$\gamma_{i,j} = 2c_{j_{\text{norm}}}(1 - t_{j_{\text{norm}}}).$$

Note that no threshold is used anymore. Instead, two new magnitudes, $c_{j_{\text{norm}}}$ and $t_{j_{\text{norm}}}$ are used. The former is the normalized processing capacity of node j, while the latter is the normalized average time spent by a search message at node j (processing time plus time in queue). The normalization of $c_{j_{\text{norm}}}$ is computed as follows. Let c_j be the capacity of node j, and $c_{\max} = \max_{k \in C_i}\{c_k\}$. Then

$$c_{j_{\text{norm}}} = \frac{c_j}{c_{\max}}.$$

Similarly, let $t_{\max} = \max_{k \in C_i}\{t_k\}$ and $t_{\min} = \min_{k \in C_i}\{t_k\}$. Then,

$$t_{j_{\text{norm}}} = \frac{t_j - t_{\min}}{t_{\max} - t_{\min}},$$

It is easy to observe that the normalized magnitudes take values in the interval $[0, 1]$. A value of $c_{j_{\text{norm}}}$ close to one means that the node j has more processing capacity than most others in C_i. Similarly, a small $t_{j_{\text{norm}}}$ implies that queries spend less time in j that in other peers in C_i. All this justifies the choices of $\gamma_{i,j}$ and $w_{i,j}$. From these values the probabilities $p_{i,j}$ are defined as in the previous section

Table 1. Capacities and upload bandwidths distribution for simulations

Capacity level	Percentage of nodes	Processing capacity c_i	Bandwidth b_i
1x	20 %	0.1	0.01
10x	45 %	1	0.1
100x	30 %	10	1
1000x	4.9 %	100	10
10000x	0.1 %	1000	100

Simulation Results. To study this second version of DANTE in a large-scale heterogeneous system, we have implemented the corresponding algorithms and tested them in a simulated system. The simulator developed allows to set the capacity of each peer by two parameters: *bandwidth* and *processing capacity*. Searching for a resource in the lists of known resources of a node i takes a time proportional to the number of resources checked m and the node's processing capacity c_i, as $t_{proc} = \frac{m}{c_i}$. The time to send a message depends on the node's bandwidth b_i and the packet size s, $t_{send} = \frac{s}{b_i}$. Processing and message transmission are assumed to overlap. Nodes capacities and bandwidths are assigned following the distribution presented in Table 1. This distribution is derived from the bandwidth distribution measured in Gnutella reported by Sariou *et al.* [12].

In the simulations each node starts a new search for a resource chosen uniformly at random periodically. The *time between searches*, tbs, is a parameter of the simulation that allows to set the load of the system. Each node holds 100 resources. All resources have the same popularity (no resource is more likely to be looked for than other). The *replication rate* r is another simulation parameter, that states the ratio of nodes that hold each resource (in percent).

Nodes manage 10 *native* connections each. Reconnections are triggered every 30 seconds of virtual time. Nodes

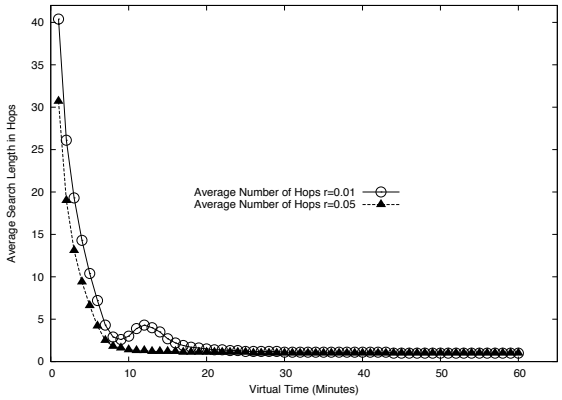

Figure 2. Number of hops to complete a search.

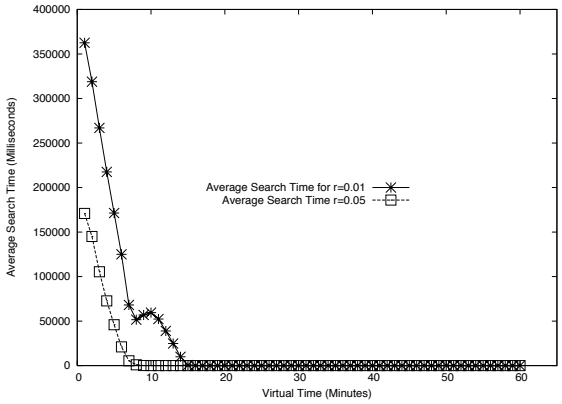

Figure 3. Time to complete a search.

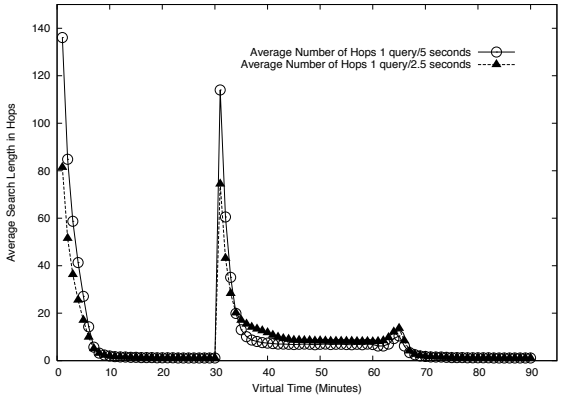

Figure 4. Number of hops to complete a search in a system under attack.

change 5 *native* connections at each reconnection. We assume that there is an external service that provides peers, at start-up time, with a list of some other nodes present in the system. When some peer is started, it chooses its initial neighbors uniformly at random from the list provided by that service. Hence, all experiments start with a random topology. Similarly, if a *native* connection points to some node that leaves the network (is attacked or deactivated), that connection is redirected to another peer chosen uniformly at random from a list obtained from the external service. Finally, the node sampling messages use a TTL of 30 while resource search messages use a TTL of 1000. Both values were chosen empirically. The first one proved to be enough to get a good sampling of the network, and the second one allowed to achieve a high success rate.

Figure 2 presents the evolution of the number of hops that are required to complete a search since the system is started. These simulations have been run for 60 minutes of virtual time with 10000 nodes, a value of tbs of one second, and two levels of replication, $r = 0.01$ and $r = 0.05$. Since the experiments start from a random topology and the system is not fully loaded, it should converge to a centralized topology. This behavior can, in fact, be observed, since the number of hops decreases rapidly as time passes, until it is stabilized at 1 after some minutes (tens of reconnections). This means that the network has reached a centralized topology, and all searches are solved in just one hop. Figure 3 shows how the topology evolution makes the average search time to decrease in the same two simulations. This figure shows that DANTE builds a very efficient topology, in which searches are completed in very little time (about 30 milliseconds). Additionally, simulations that explore the behavior of DANTE under churn have been presented in [11].

To conclude this section, we present Figures 4 and 5, which show how the system behaves under a targeted attack. Since at the simulated load levels the system converges to a centralized topology, the attack will remove the central nodes of the topology. The simulations presented here were run with 10000 nodes, and replication $r = 0.01$. Two different loads were tested. Each simulation lasts for 90 minutes of virtual time. At minute 30, when the network has moved to a starlike topology, an attack is performed: the 10 best connected nodes (central nodes) are forced to leave the network. Those are also the 10 nodes with the greatest capacities (see Table 1). 30 minutes later, those nodes are back in the system.

Figure 4 shows how the average number of hops to find resources decreases sharply in a few reconnections until it reaches a value close to 1. At that moment the network has a starlike topology. When the attack is performed at minute 30, the remaining nodes redirect their connections randomly, so a random topology appears again. From that moment on, nodes will try to connect to the remaining peers with highest capacity (in this case, nodes of the fourth Capacity Level at Table 1). The network does not become centralized, since these nodes do not have enough capacity to become central. Yet, highly connected nodes appear,

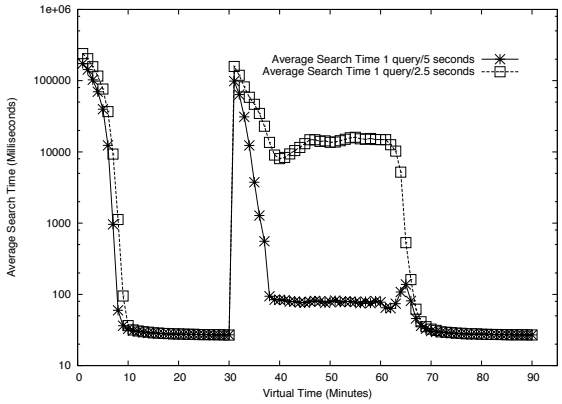

Figure 5. Time to complete a search in a system under attack.

so the average number of hops decreases sharply in a few minutes. Finally, when the high capacity nodes are back, the network evolves again to a starlike topology.

In Figure 5 we see how the attack affects the search completion times. As expected, those times increase to values close to those obtained at the beginning of the experiment. Then, as nodes adapt their connections, the topology is adapted again, lowering the average search time to a fair value. Finally, when the 10 nodes attacked are back, the search times gradually return to the values before the attack.

We can conclude that DANTE can be temporarily affected by well-targeted attacks. However, even in a scenario where nodes that have become central are all successfully attacked at the same time, and no other nodes of the same capacity remain in the system, the network adapts again to reach another efficient state. The system is never fully shut down, because it is not dependant on any particular subset of nodes.

5 Future Work

The results presented seem to indicate that the combination of random walks and self-organizing overlay networks may lead to very efficient unstructured P2P systems. However, we believe that more work has to be invested to fully understand the behavior of these systems and hence to be able to fine tune them.

Among the problems we are currently studying, a very important one related to our reconnection mechanisms is when and how to reconnect. The reader possibly noticed that from the first to the second version of DANTE we moved form changing all the native connections at once to change only half of them. That has given some stability to the networks, which has proven to be beneficial. We plan to explore other ways to do this reconnection. Regarding when to reconnect, we have only explored periodic reconnections, but it is natural to ask whether a different approach, like state-driven reconnections, could be more interesting.

Another of our current lines of work is to identify the right values for the different TTLs used. For that, we have done an analysis of the average length of a random walk in a network of given degree distribution. Hopefully we should be able to use this analysis to tune the values of the TTLs and, maybe, even the topology of the network.

Finally, we plan to explore alternatives to pure random walks, like biased random walks (that, for instance, have a preference for neighbors with high degree), or random walks that try to avoid revisiting the same nodes again. We have high hopes in these alternatives.

References

[1] L. A. Adamic, B. A. Huberman, R. M. Lukose, and A. R. Puniyani. Search in power law networks. *Physical Review E*, 64:46135–46143, October 2001.

[2] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, December 2004.

[3] Y. Chawathe, S. Ratnasamy, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *SIGCOMM 2003*, pages 407–418, Karlsruhe, Germany, August 2003.

[4] V. Cholvi, V. Laderas, L. López, and A. Fernández. Self-adapting network topologies in congested scenarios. *Physical Review E*, 71(3):035103, 2005.

[5] G. H. L. Fletcher, H. A. Sheth, and K. Borner. Unstructured peer-to-peer networks: Topological properties and search performance. In *AP2PC 2004*, New York, New York, United States, July 2004.

[6] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *INFOCOM 2004*, volume 1, pages 120–130, Hong Kong, March 2004.

[7] R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, and A. Arenas. Optimal network topologies for local search with congestion. *Physical Review Letters*, 89, November 2002.

[8] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, New York, United States, June 2005.

[9] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make Gnutella scalable? In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 94–103, Cambridge, United States, March 2002.

[10] L. Rodero-Merino, L. López, A. Fernández, and V. Cholvi. Dante: A self-adapting peer-to-peer system. In *AP2PC 2006*. Springer-Verlag, 2006.

[11] L. Rodero-Merino, L. López, A. Fernández, and V. Cholvi. A topology self-adaptation mechanism for efficient resource location. In *ISPA 2006*, pages 660–671. Springer-Verlag, 2006.

[12] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *MMCN'02*, volume 4673, pages 156–170, 2002.