A Topology Self-adaptation Mechanism for Efficient Resource Location[⋆]

Luis Rodero-Merino[1], Luis López[1], Antonio Fernández[1], and Vicent Cholvi[2]

[1] LADyR, Universidad Rey Juan Carlos, 28933, Móstoles, Spain
{lrodero, llopez, anto}@gsyc.escet.urjc.es
[2] Universitat Jaume I, 12071, Castellón, Spain
vcholvi@lsi.uji.es

Abstract. This paper introduces a novel unstructured P2P system able to adapt its overlay network topology to the load conditions. The adaptation is performed by means of a mechanism which is run by the nodes in the network in an autonomous manner using only local information, so no global coordinator is needed. The aim of this adaptation is to build an efficient topology for the resource discovery mechanism performed via *random walks*. We present the basis of the adaptation mechanism, along with some simulation results obtained under different conditions. These results show that this system is efficient and robust, even in front of directed attacks.

1 Introduction

The Peer-to-Peer (P2P) paradigm has brought new communication opportunities for Internet users. P2P systems present advantages like flexibility, scalability and fault tolerance, thanks to the lack of central coordinators or controllers. But this same lack of central entities has brought new technical challenges.

Maybe one of the key issues to be solved is how to locate resources efficiently. Several solutions have been proposed, each with its advantages and drawbacks. It seems that P2P systems with *unstructured overlay networks* are suitable for certain scenarios like mass-market distributed resource sharing.

Unfortunately, it is not trivial to offer efficient search solutions in unstructured networks. Flooding based proposals (like the first versions of Gnutella) present scalability issues. Because of this, the research community is making efforts to develop solutions based on *random walks*. More specifically, to combine random walks with *dynamic overlay topologies* (topologies that change during the system life) is an approach that has lead to promising results [1, 2].

Here we introduce a solution based on the same idea of using random walks along with a dynamic topology. Changes on the topology are performed by the nodes themselves to adapt it to the load on the network. In order to do this, each node runs a

[⋆] This work was partially supported by the Spanish Ministry of Science and Technology under Grant No. TSI2004-02940 and TIN2005-09198-C02-01, by Bancaixa under Grant No. P1-1B2003-37 and by the Comunidad de Madrid under Grant No. S-0505/TIC/0285. The authors would like to thank Gia creators Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham and Scott Shenker for providing us with their simulator source code.

reconnection mechanism (the same for all nodes) that periodically computes to which other peers the node must connect. This solution has been implemented in a system called **DANTE$_2$**[1].

This paper is organized as follows. In Sect. 2 we revise some current solutions for resource location in P2P networks. Section 3 presents previous works which form the conceptual basis of DANTE$_2$. Section 4 describes the adaptation mechanism used by DANTE$_2$. Section 5 presents some results obtained by simulation. Those results measure the performance of DANTE$_2$ under a variety of circumstances. Finally, Sect. 6 contains the conclusions of this paper and possible lines of future work.

2 Resource Discovery in P2P Systems

Traditionally, solutions for resource location in P2P systems are classified in two groups: *centralized* and *decentralized*. In centralized solutions a central repository stores an index of all resources in the network (like in Napster [3]). This approach makes the system vulnerable to attacks or censorship and poses scalability issues.

In decentralized systems, on the other hand, the resource discovery service is provided by the peers themselves. Decentralized systems are usually classified by the kind of *search mechanism* they implement to route search messages through the network:

1. *Structured systems*. These systems use specialized placement algorithms to assign responsibility for each resource to specific peers, as "directed" search mechanisms to efficiently locate resources. One example is Chord [4].
2. *Unstructured systems*. These systems do not have precise control over the resource placement and, traditionally, use search mechanisms based on flooding, random walks or supernodes. Examples are Gnutella and KaZaA.

Structured systems are very efficient: they usually require few communication steps to find some resource, and do not produce false negatives (i.e., the search fails only if the demanded resource is not in the system). On the other hand, unstructured systems tend to be less efficient, and may yield false negatives.

Yet, unstructured systems have some advantages: they have little management overhead, adapt well to the transient activity of P2P nodes, take advantage of the spontaneous replication of popular content and allow to perform queries by keyword in a simpler way than with directed search protocols. These advantages seem to make unstructured systems suitable for many real-world situations, like massive file sharing systems. Further discussion comparing structured and unstructured systems can be found in [5].

2.1 Resource Location in Unstructured P2P Networks

Three search methods are typically used for resource location in unstructured networks. The first one is *flooding*, where each peer broadcasts the queries to all its neighbors. It is well known [6] that the flooding solution presents problems of scalability.

[1] From Dynamic self-Adapting Network TopologiEs.

Another solution is based on the idea of *superpeers*. These are special nodes that store the index of resources shared by the rest of peers. The eDonkey [7] network, for instance, uses this approach. Yet, these systems are dependant on the availability of those powerful enough nodes.

Finally, the third search mechanism is based on *random walks*. Here, nodes forward each query to only one peer, chosen randomly among its neighbors. There is little communication overhead compared with flooding, but it can take longer to solve queries. In [8] and [9] we can find some studies that state that random walks seem to be a promising technique suitable to solve the scalability problems of flooding.

2.2 Dynamic Topologies Based Proposals

It is well known that the performance of random walks is highly dependant on the topology of the overlay network [10, 11, 12]. Thus, some solutions have been proposed that try to adapt the overlay topology to the network load, in order to improve the efficiency of the search process.

First, Lv *et al.* [2] introduced a P2P system in which nodes avoid congestion by means of a flow control mechanism that redirects the most active connections to neighbors with spare capacity. Another work is **Gia** [1], proposed by Chawathe *et al.* In Gia, queries are forwarded to high capacity nodes. An active flow control mechanism avoids overloading hot spots: each node notifies its neighbors the number of queries they can send to it, which depends on its spare capacity. Topology is adapted by a mechanism based on nodes' *level of satisfaction*, which measures the distance between a node's capacity to the sum of its neighbors capacities, normalized by their degrees. This parameter determines whether or not each node will adapt the topology, and the frequency of these adaptations.

DANTE$_2$ implements a different reconnection mechanism that we deem can lead to even more efficient topologies. First of all, DANTE$_2$ is inspired on the results of Guimerà *et al.* [13] on the relationship between network topologies and search performance (see Sect. 3). Another difference is that nodes in DANTE$_2$ do not keep track of their neighbors' state, nor implement any explicit flow control technique. Thus DANTE$_2$ avoids the communication overhead due to those activities. Some simulations comparing DANTE$_2$ and Gia are presented in Sect. 5.

3 Previous Work

The self-adaptation mechanism used in DANTE$_2$ is inspired on the results of Guimerà *et al.* [13] and on the algorithm proposed by Cholvi *el al.* [10]. Guimerà *et al.* were able to characterize the topologies that, given a search mechanism based on random walks, minimize the average time needed to perform a search. They found that when the system is not congested, the optimal topology is a star-like structure. Furthermore, they also found that when the system is congested, the optimal topology is a random-like one. However, Guimerà *et al.* did not state how these topologies could be achieved dynamically in a real system. In P2P networks, we face two problems: the lack of global knowledge, and the absence of a coordinator that tells nodes to which other peers they must connect to.

Thus, applying Guimerà results to P2P networks is not straightforward. A topology adaptation mechanism that fits P2P systems should be run locally at the nodes, and should not need global knowledge. Cholvi *et al.*, in [10], proposed a first mechanism that, depending on the current system load, makes nodes to locally change their connections so that the obtained topologies are random for high loads and star-like for low loads. Yet, in that solution nodes need to now all other peers state.

Finally, both in [13] and in [10] it is assumed that all participants have the same capacities, that is, the network is *homogeneous*. Nonetheless, nodes in real networks are known to be *heterogeneous* [14].

A previous version of DANTE$_2$ was introduced in [15]. Although the core idea of that work was the same used here (using a self-adapting topology to improve searches efficiency), this paper introduces key improvements: a better, more accurate reconnection mechanism that takes into account the nodes heterogeneity, and more complete simulations in different and more realistic scenarios.

4 DANTE$_2$ Self-adaptation Mechanism

In DANTE$_2$ each peer knows its own resources as well as the resources held by its neighbors. Based on this, it is easy to understand that nodes are more interested on being connected to peers with many neighbors. Therefore, DANTE$_2$ encourages peers to establish connections with high degree nodes. DANTE$_2$, in fact, aims to form highly clustered (even centralized) topologies. However, this holds only as long as highly connected nodes can handle all the incoming traffic.

Taking into account this reasoning, DANTE$_2$ uses an algorithm that, when the network traffic is low, drives the network to a star-like overlay topology. Thus, searches can be answered in only one hop, since the central nodes know all the resources in the system. In turn, when the number of searches increases, well-connected nodes will become congested and their neighbors will start to disconnect from them. Hence, this will drive the network to a more random-like topology that, although it makes search messages to traverse longer paths to find some resource, will balance the load and perform better than by using a highly congested central node.

More specifically, in DANTE$_2$ each node can establish connections to other nodes. We say that a connection is *native* for the establishing node and *foreign* for the accepting node. Nodes can change their native connections, but not their foreign ones. Furthermore, each node periodically runs a reconnection mechanism with which native connections are changed. This mechanism firstly obtains a list of potential candidates C to which it can connect. Then, it assigns a probability p_i to each candidate i, and chooses candidates at random using their respective probabilities. Finally, the peer reconnects its native connections to the chosen candidates.

The probability assigned to a candidate $i \in C$ is based on its "attractiveness", denoted as Π_i and defined as

$$\Pi_i = k_i^{\gamma_i} \tag{1}$$

where k_i is the degree (number of neighbors) of peer i, and γ_i is computed as

$$\gamma_i = 2 \times c_{i_{norm}} \times (1 - t_{i_{norm}}) \tag{2}$$

$c_{i_{norm}}$ is the normalized processing capacity of node i, where the normalization is performed as follows. Let c_i be the capacity of node i, and $c_{max} = \max_{i \in C}\{c_i\}$. Then,

$$c_{i_{norm}} = \frac{c_i}{c_{max}} \tag{3}$$

it follows that $0 < c_{i_{norm}} \leq 1, \forall\, i$, where a larger $c_{i_{norm}}$ means that node i is more attractive as it has more capacity to process searches.

$t_{i_{norm}}$ represents the average time spent by a search at node i (time in queue plus processing time), normalized. The normalization is computed as follows. Let t_i be the mean search processing time of node i, $t_{max} = \max_{i \in C}\{t_i\}$ and $t_{min} = \min_{i \in C}\{t_i\}$. Then,

$$t_{i_{norm}} = \frac{t_i - t_{min}}{t_{max} - t_{min}} \tag{4}$$

It is straightforward to see that $0 \leq t_{i_{norm}} \leq 1 \,\forall\, i$, where a lesser $t_{i_{norm}}$ means that node i is more attractive as it takes less time for searches to be served.

Finally, once the Π_i values are computed for all candidates in C, each candidate $i \in C$ is assigned a probability p_i of being chosen that is computed as

$$p_i = \frac{\Pi_i}{\sum_{j \in C} \Pi_j} \tag{5}$$

By the definition of Π_i (Eq. 1), the attractiveness of node i is strongly dependant on its degree k_i. The higher the degree, the more attractive the node becomes, and so more peers will try to connect to it, increasing again k_i (and therefore Π_i). This process leads quickly to centralized topologies. The form of γ_i, on the other hand, comes from the fact that the reconnection function must favor high capacity nodes (capacity is represented by c_i), and avoid loaded peers (load is given by t_i). Thanks to this reconnection mechanism, the system behaves in an adaptative manner, changing its topology to suit the load conditions.

Candidates Sampling. The reconnection mechanism of DANTE$_2$ depends on the set of candidates C to which the node can connect. There are several mechanisms that could be used to build this list of candidates. For example, a *gossiping* based service like [16] could spread information about nodes in the network. Another solution is to make nodes to keep a *cache* of other peers in the network.

DANTE$_2$ implements a third solution. Whenever a node starts a new reconnection, it launches a special *Look For Node* message, that traverses the network following a random walk with a bounded TTL. When the TTL expires, another message is sent to the source node with the list of traversed peers. This list becomes the set of candidates. This technique has small incidence on the network load, and Newman's results [17] show that the set obtained is a good sample of the overall network.

5 Simulations

To study DANTE$_2$'s performance we have developed a simulator that implements its reconnection mechanism. Simulations use the microsecond as the minimum unit of time.

The capacity of each node is set by two parameters: *bandwidth* and *processing capacity*. Nodes perform tasks, like the processing of an incoming message or an internally started process (e.g., the triggering of a new reconnection). When performing some task the node is said to be *busy*. Any other pending task in the node is enqueued until the present task is finished.

The processing time t_{proc} depends on the tasks being performed. Tasks other than searching for a resource in the lists of known resources are assumed to take one unit of time. Searches for resources take a time proportional to the number of resources checked m and the node's processing capacity c_i, $t_{proc} = \frac{m}{c_i}$. Some tasks need to send a message. The duration of sending a message , t_{send}, depends on the node's bandwidth b_i and the packet size s, $t_{send} = \frac{s}{b_i}$. Finally, the time the node is busy, t_{busy}, for one task is computed as $t_{busy} = \max\{t_{proc}, t_{send}\}$. This time is not $t_{proc} + t_{send}$, because we assume that the sending of messages and the processing of searches run in a pipeline. Nodes capacities and bandwidths are assigned following the distribution depicted in Table 1. This distribution is derived from the measured bandwidth distributions of Gnutella nodes reported in [14].

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

Each node starts a new search for a random resource periodically. The *time between searches*, *tbs*, is a parameter of the simulation that allows to set the load on 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 rate of nodes that hold each resource (in percentage).

Nodes manage 10 *native* connections each. Reconnections are triggered every 30 seconds of virtual time. Nodes change 5 *native* connections at each reconnection. We assume 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 randomly 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 at random from a list again obtained from the external service.

The *Look For Nodes* messages have a TTL of 30 hops. Resource search messages have a TTL of 1000 hops. Both values were chosen empirically. The first one proved to be enough to get a good sampling of the network, the second one allowed to obtain a high success rate, both for DANTE$_2$ and Gia simulations.

Topology Evolution in DANTE$_2$. First, we study how in DANTE$_2$ the system is able to adapt itself, changing its topology as the (virtual) time passes. The results of two simulations are shown, with two different replication rates: $r = 0.01$ and $r = 0.05$. Both simulations are run with 10000 nodes (so $r = 0.01$ implies that each resource is held by only one node) and a time between searches $tbs = 1$ second. Simulations were run for 60 minutes of virtual time. All searches finished successfully in both simulations.

In Fig. 1.(a) we see how the mean number of hops changed as the virtual time passed. In the X axis we represent the virtual time, in minutes. In the Y axis we represent the average number of hops that took to solve searches started during the corresponding minute of virtual time. The number of hops decreases readily as the time passed, until it is stabilized to 1 after some minutes (tens of reconnections). This means that the network has reached a centralized topology, starting from a random one, and all searches are solved in just one hop.

On the other hand, we see in Fig. 1.(b) how the topology evolution makes the average search time to decrease. DANTE$_2$ builds an efficient topology, where searches are completed in very little time (about 30 milliseconds).

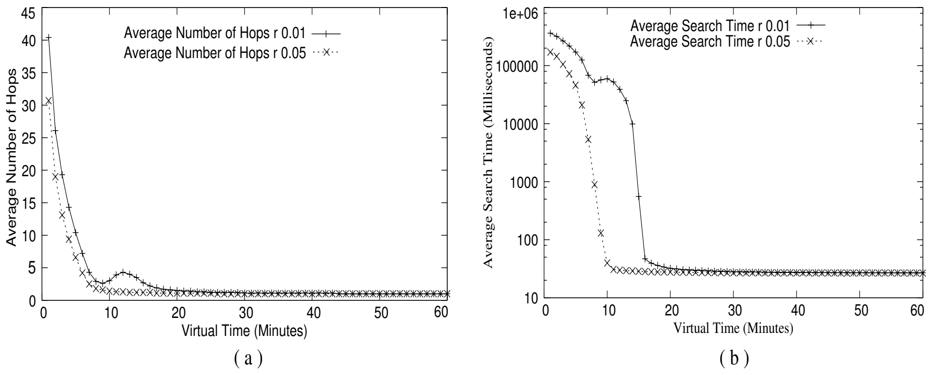

Fig. 1. Average number of hops and searches duration

Robustness Against Peers Churn. It is well known that peers may enter and leave the network at a high rate. This can, in some cases, compromise the efficiency of the system. In this section we present some simulations results that show how DANTE$_2$ performance is not strongly affected by the churn of peers.

The simulations of this section are run with 10000 nodes. The replication is $r = 0.05$ and the time between searches is $tbs = 5$ seconds. Only searches started between minutes 31 and 60 (included) are taken into account for the results. Searches started before minute 31 are discarded to avoid the initial transition state. The simulations were run until all searches started before minute 61 were finished.

Initially, nodes are active with a probability of 0.5. At start time, active nodes form a random topology. Each active node will run for a certain time that is independently calculated using a exponential distribution. When the time expires, the node is deactivated: it discards all searches in its queue, and closes its connections with all its neighbors.

After 0.5 seconds of virtual time, the node changes to the active state again, pointing its *native* connections to 10 nodes chosen at random, and the time to remain active is recalculated. This is similar to simulate nodes leaving the system as other nodes simultaneously joining it (a similar approach is used in [1]).

To simulate different churns, we set different values for the mean of the exponential distribution used to compute the time nodes will stay active: 1, 5, 10, 50 and 100 minutes. Finished searches are classified into three categories: *successful*, *failed* (the search TTL expired before the resource was found) or *discarded* (search was at a node that changed to the deactivated state). Figure 2 shows searches results for each experiment.

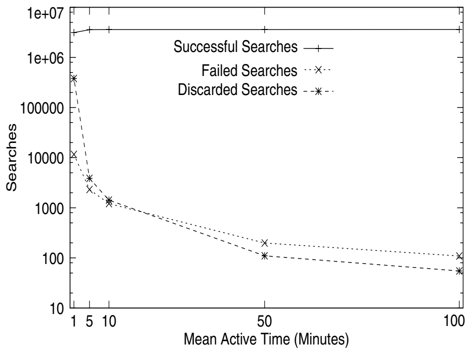

Fig. 2. Search results under churn

For the higher churn, the number of discarded plus failed searches is about 10% of the total. Yet, when the mean of the distribution increases, the number of discarded and failed searches decreases sharply. When the mean is 10 minutes, the number of unsuccessful searches (approximately 2600) represents only the 0.07% of the total.

In Figs. 3.(a) and 3.(b) it is shown how the churn of peers affects the search performance (only for successful searches). Although the number of hops and the time to complete queries increase, they are within what we deem are acceptable bounds even when the churn of peers is high.

Robustness Against Attacks and Congestion Avoidance. In DANTE$_2$, high capacity nodes tend to have more connections, even forming a starlike topology. Thus, it can be argued that DANTE$_2$ is vulnerable to attacks targeted to well-connected nodes, or that those nodes could become congested and so compromise the system performance. In this section we discuss how attacks can affect DANTE$_2$'s behavior, and how congestion is avoided by the reconnection mechanism. The simulations presented here were run with 10000 nodes, and replication $r = 0.01$. Two different loads were tested.

The results are shown in Figs. 4.(a) and 4.(b). Both of them show how the network behavior evolves as the virtual time passes, from the first minute of simulation (remember that initially nodes form a random topology) to minute 90. At minute 30, when the network has moved to a centralized topology, an attack is performed: the 10 best connected nodes (central nodes) are forced to leave the network. Those are also the

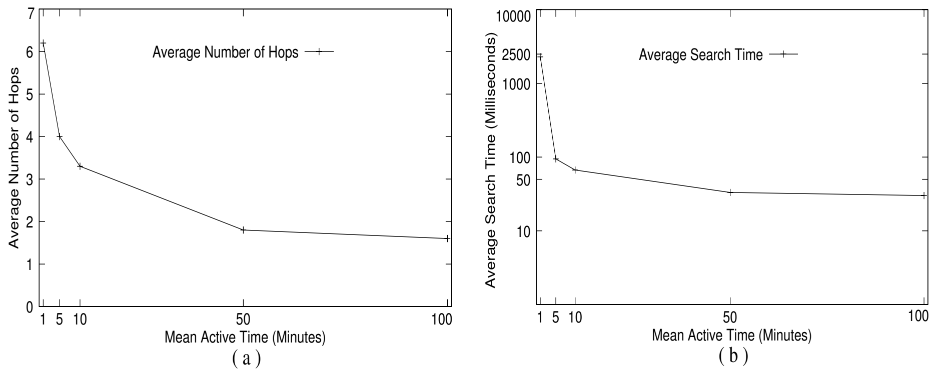

Fig. 3. Searches hops and duration under churn

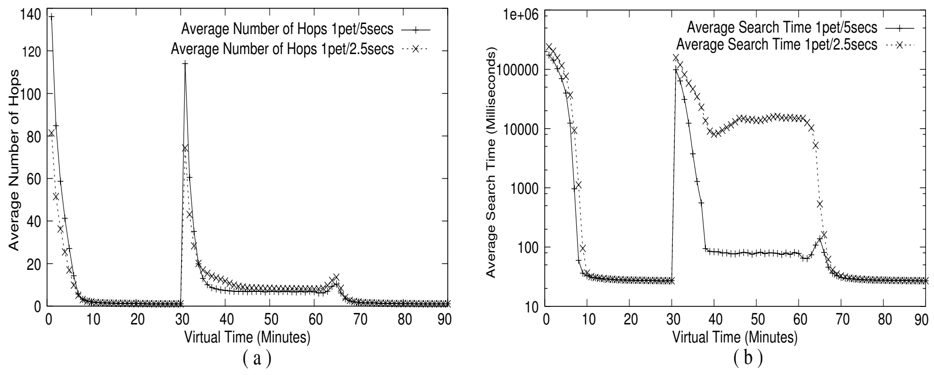

Fig. 4. Searches hops and duration under attack

10 most capable nodes in the network (see Table 1). 30 minutes later, those nodes are reactivated. We will check how DANTE$_2$ reacts to those events.

Figure 4.(a) 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 centralized 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 higher capacity (in this case, nodes of the fourth Capacity Level at Table 1). The network is never centralized again, as there are no nodes with enough capacity to become central. Here we can see how DANTE$_2$ actively avoids overloading nodes, not allowing them to receive too many connections if they can not handle them. Yet, highly connected nodes appear so the mean number of hops decreases sharply in a few minutes (to lesser values with lesser load). Finally, when the attacked nodes are back, the network changes again to a centralized topology.

In Fig. 4.(b) we see how the attack affects to the search times. As expected, those times increase to values close to those obtained at the beginning of the experiment.

Then, as nodes change their connections the topology is adapted again, lowering the average search time to a fair value. It can also be observed that the network has reached again a stable state, due in part to the fact that no node gets overloaded. If well connected nodes had become congested, then their messages queues would grow indefinitely, and so would the resulting average searches times. Finally, when the 10 nodes attacked are back, the search times gradually return to the values previous to the attack. In both experiments, the proportion of discarded searches is around 0.005%, and the proportion of failed searches is less than 0.04%.

We can conclude that DANTE$_2$ can be temporarily affected by well targeted attacks. Yet, 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.

These experiments also show how the reconnection mechanism implemented by DANTE$_2$ avoids overloading peers, preventing the network from reaching an unstable state. If there are not enough high capacity nodes that allow to form a centralized configuration, the resulting topology becomes more 'randomized'. In conclusion, DANTE$_2$ maintains at all times the topology as clustered as possible, but at the same time prevents nodes from becoming overloaded.

DANTE$_2$ vs. GIA. As explained in Sect. 2.2, Gia is another proposal of a P2P system that uses an adaptation mechanism to improve the efficiency of searches. In [1] Gia authors carried on some simulations that show how self-adapting networks can offer a better performance than other solutions (like flooding) in a variety of scenarios. Thus, instead of repeating those same simulations with DANTE$_2$, we have deemed more interesting to compare Gia and DANTE$_2$.

We have developed a Gia simulator that implements the mechanisms described in [1]: a *flow control* system to avoid overloading nodes, a *biased random walk* search mechanism, and a *topology adaptation* protocol. An important parameter of Gia is the *maximum number of neighbors* (*max_neigh*). Nodes in Gia try to connect to as many nodes as possible, and to those with the highest capacity. We have set that limit to 20 (twice the number of *native* connections in DANTE$_2$), so that the average degree is the same as the one obtained in the DANTE$_2$ simulations. Gia advocates could reason that setting a higher maximum bound would improve performance, as searches would need less hops to locate resources. But then, it would be enough in DANTE$_2$ to increase the number of nodes $native$ connections to $max\_neigh/2$ again.

Simulations were run with 1000 nodes, with replication $r = 0.1$. As usual, nodes capacities and bandwidths are set following the distribution of Table 1. Only searches started between minutes 31 and 60 are taken into account.

In Fig. 5.(b) we plot the average search times for different loads on both systems. DANTE$_2$ seems to perform better than Gia for all loads. Additionally, beyond a certain point, Gia search times start to grow quickly with the system load, while DANTE$_2$ is able to keep search times low for the same loads. Figure 5.(a) helps us to understand the reason of DANTE$_2$'s better behavior: searches in Gia need many more hops to find a certain resource (about 160) than in DANTE$_2$ (about 7). The reason is that, although the topology in DANTE$_2$ is not totally centralized (as there are not enough high capacity

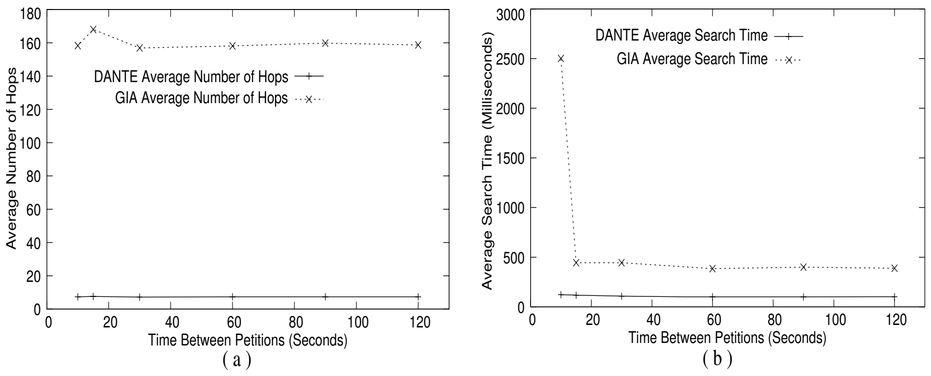

Fig. 5. DANTE and GIA searches number of hops and duration

nodes), it still keeps a clustered form where a few nodes are well connected and hence allows queries to be completed in a few hops.

All searches were successful in DANTE$_2$. Gia, on the other hand, presented a certain proportion of failed searches (between 1.5% and 2%) in all experiments.

6 Conclusions and Future Work

P2P systems are a promising new paradigm, yet they demand innovative solutions to new problems like resource location. DANTE$_2$ proposes a self-adapting mechanism that makes the network change its topology aiming always to an efficient configuration that depends on the system load and the peers capacities. The results obtained with DANTE$_2$ seem promising. However, much work remains to be done in order to improve the performance of these techniques. For example, new reconnection heuristics could be studied and developed.

References

1. Chawathe, Y., Ratnasamy, S., Lanham, N., Shenker, S.: Making Gnutella-like P2P systems scalable. In: Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2003), Karlsruhe, Germany (2003) 407–418
2. Lv, Q., Ratnasamy, S., Shenker, S.: Can heterogeneity make Gnutella scalable? In: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Cambridge, United States (2002) 94–103
3. The napster website http://www.napster.com.
4. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2001), San Diego, CA, United States (2001) 149–160
5. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys **36** (2004) 335–371

6. Ritter, J.: Why gnutella can't scale. no, really. (Technical report) Electronic format in http://www.darkridge.com/ jpr5/doc/gnutella.html.
7. The edonkey and overnet website http://www.edonkey2000.com.
8. Fletcher, G.H.L., Sheth, H.A., Borner, K.: Unstructured peer-to-peer networks: Topological properties and search performance. In: Proceedings of the Third International Workshop on Agents and Peer-to-Peer Computing, New York, New York, United States (2004) To be published by Springer.
9. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th international conference on Supercomputing, New York, New York, United States (2005) 84–95
10. Cholvi, V., Laderas, V., López, L., Fernández, A.: Self-adapting network topologies in congested scenarios. Physical Review E **71** (2005) 035103
11. Adamic, L.A., Huberman, B.A., Lukose, R.M., Puniyani, A.R.: Search in power law networks. Physical Review E **64** (2001) 46135–46143
12. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2004. Volume 1., Hong Kong (2004) 120–130
13. Guimerà, R., Díaz-Guilera, A., Vega-Redondo, F., Cabrales, A., Arenas, A.: Optimal network topologies for local search with congestion. Physical Review Letters **89** (2002)
14. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of SPIE (Proceedings of Multimedia Computing and Networking 2002, MMCN'02). Volume 4673. (2002) 156–170
15. Rodero-Merino, L., López, L., Fernández, A., Cholvi, V.: Dante: A self-adapting peer-to-peer system. In: Lecture Notes in Computer Science (Proceedings of AP2PC 2006, to appear), Springer-Verlag (2006)
16. Jelasity, M., Guerraoui, R., Kermarrec, A.M., van Steen, M.: The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In: Lecture Notes in Computer Science (Proceedings of Middleware 2004). Volume 3231., Springer-Verlag (2004) 79–98
17. Newman, M.E.J.: A measure of betweenness centrality based on random walks. Social Networks **27** (2005) 39–54