Fast Information Retrieval in a Self–Organising P2P Network

Vincenza Carchiolo, Michele Malgeri, Giuseppe Mangioni, Vincenzo Nicosia
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
Facoltà di Ingegneria - Università di Catania
V.le A. Doria, 6 - 95125 Catania (ITALY)
{car, mmalgeri, gmangioni, vnicosia}@diit.unict.it

***Abstract*— In this paper we present a formal description of *PROSA*, a P2P resource management system heavily inspired by social networks. Social networks have been deeply studied in the last two decades in order to understand how communities of people arise and grow. It is a widely known result that networks of social relationships usually evolves to small–worlds, i.e. networks where nodes are strongly connected to neighbours and separated from all other nodes by a small amount of hops. This work shows that algorithms implemented into *PROSA* allow to obtain an efficient small–world P2P network. We also show how taking advantage of *PROSA* structure it is possible to effectively answer queries. In particular, the so–called query recall for *PROSA* is estimated and compared to that obtained in SETS [1] and GES [2].**

Keywords: *P2P, Social networks, Emerging structures*

I. Introduction

A Peer-to-Peer system consists of computing elements that are connected by a network, addressable in a unique way, and sharing a common communication protocol. All computing elements, equivalently called nodes or peers, have the same functions and role. In P2P networks there is no difference between ”client” and ”server” hosts: a peer acts as a ”client” if it requests a resource from the network, while it acts as a ”server” if it is requested a resource it is sharing. From this point of view, P2P networks differ a lot from World Wide Web, TCP/IP networks and, in general, from client-server networks.

Studies on P2P networks are focused on two different topics: physical P2P networks (i.e.,P2P networks opposed to hierarchic and centralised TCP/IP networks) and overlay networks (i.e. networks of logical links between hosts over an existing physical network of any type). Our interest is mainly focused on overlay P2P systems: they are probably going to become the most used kind of application–level protocols for resource sharing and organisation.

In this paper we present a novel P2P overlay network, named ***PROSA*** [3] [4], heavily inspired by social networks. Social networks are sets of people or groups interconnected by means of acquaintance, interaction, friendship or collaboration links. Many kinds of natural social networks have been deeply studied in the last thirty years [5], and many interesting characteristics of such networks have been discovered. In a real social network relationships among people are of the most importance to guarantee efficient collaboration, resources discovery and fast retrieval of remote people. Nevertheless, not all relationships in a social network are of the same importance: usually links to parents and relatives are stronger than links to friends, which are in turn stronger than links to colleagues and class mates. On the other hand, it is also interesting to note that usually links in a social group evolve in different ways. A large amount of relationships are (and remain) bare “acquaintances”; some of them evolve around time into “friendships”, while “relativeness” is typical of very strong links to really trusted people.

This suggests that a P2P network based on a social model should take into account that different kind of links among peers can exist, and that links can evolve from simple acquaintances to friendship.

Results of studies performed by Watts, Strogatz, Newman, Barabasi et al. in the last decades[6] [7] [8] [9] [10] reveal that networks of movie characters, scientific collaborations, food chains, proteins dependence, computers, web pages and many other natural networks usually exhibit emerging properties, such that of being small–worlds. A small–world is a network where distance among nodes grows as a logarithmic function of the network size and similar nodes are strongly connected in clusters.

PROSA tries to build a P2P network based on social relationships, in the hope that such network could naturally evolve to a small–world.

In section III, we describe ***PROSA*** and algorithms involved in linking peers and routing queries for resources; in section IV, we discuss topological properties of ***PROSA*** network, obtained by simulation; section V reports some results about ***PROSA*** performance in searching and retrieving resources; in section VI, we summarise obtained results and plan future work.

II. Related Works

In the last decade researchers have proposed many models for P2P overlay networks, the most common being “Unstructured Overlays” and “Distributed Hash Tables” (DHT) [11]

Unstructured Overlay networks are the simplest P2P networks. Each node is directly connected to a relatively

small amount of other nodes. Queries for resources are usually based on limited flooding, i.e. the query is forwarded to all connected nodes, excepted the one who sent it, provided that the same query is not forwarded twice from the same node, and that queries live just for a limited number of "hops". The most famous example of such a network is Gnutella [12]. The worst aspect of unstructured P2P networks is that flooding is not efficient, and limited flooding does not guarantee to find matching results.

DHT–based P2P, on the other hand, are able to give a certain internal structure to the network. Nodes are usually assigned unique ID and are connected to a certain number of nodes that results to be "near to them" according to a given hashing function and to a similarity index. Queries are usually routed using the best available link and the underlying structure allow to forward queries in a really efficient way, usually in O($\log^k n$) hops, where n is the number of nodes in the network and k is a small integer between 1 and 3. The main problem with DHTs is that resources have to be queried using their ID, which is not a human readable string. No query "a–la–google" are allowed in such kind of networks, and they result to be not so usable, indeed.

Some recent works [1][2] focused their interest on introducing a certain amount of semantic in P2P overlays, allowing query to become more readable and understandable by users, while trying to maintain good performance in terms of resource availability, recall and robustness.

In particular, SETS [1] uses a hybrid P2P network where all nodes are spread into a predefined number of clusters (topic segments), depending on the kind of resources they are sharing, and a network manager (a super-peer) in responsible of periodically reassigning nodes to each cluster in order to maintain consistency. The main issue of SETS is that the reliability and performance of the network is devoted to a single super-peer, which represents a possible point of fault. On the other hand, the number of clusters is an arbitrary value and it is not clear how it impacts on performance.

A completely different approach is used in GES [2]. No super-peer is in charge of creating clusters of similar nodes and network management is completely distributed. Each peer decides to link some other peer depending on a similarity index, after a handshake phase. Queries are forwarded using a hybrid algorithm: if a peer does not have links to relevant nodes, it forwards the query along a random path. Otherwise, the most relevant neighbour is selected as next hop. The matter with GES is that network management requires additional messages to be exchanged among neighbours. As in SETS, also in GES resources are represented by a Vector Space Model[13], in terms of vectors of TF–IDF coefficients.

III. ***PROSA***: A BRIEF INTRODUCTION

As stated above, ***PROSA*** is a P2P network based on social relationships. More formally, we can model ***PROSA*** as a directed graph:

$$\textbf{\textit{PROSA}} = (\mathcal{P}, \mathcal{L}, P_k, Label) \quad (1)$$

$\mathcal{P}$ denotes the set of peers (i.e. vertexes), $\mathcal{L}$ is the set of links $l = (s, t)$ (i.e. edges), where t is a neighbour of s. For link $l = (s, t)$, s is the source peer and t is the target peer. All links are directed.

In P2P networks the knowledge of a peer is represented by the resources it shares with other peers. In ***PROSA*** the mapping $P_k : \mathcal{P} \rightarrow \mathcal{C}$, associates peers with resources. For a given peer $s \in \mathcal{P}$, $P_k(s)$ is a compact description of the peer knowledge (PK - *Peer Knowledge*).

Relationships among people are usually based on similarities in interests, culture, hobbies, knowledge and so on. Usually these kind of links evolve from simple "acquaintance–links" to what we called "semantic–links". To implement this behaviour three types of links have been introduced: *Acquaintance–Link* (AL), *Temporary Semantic–Link* (TSL) and *Full Semantic–Link* (FSL). TSLs represent relationships based on a partial knowledge of a peer. They are usually stronger than ALs and weaker than FSLs.

In ***PROSA***, if a given link is a simple AL, it means that the source peer does not know anything about the target peer. If the link is a FSL, the source peer is aware of the kind of knowledge owned by the target peer (i.e. it knows the $P_k(t)$, where $t \in \mathcal{P}$ is the target peer). Finally, if the link is a TSL, the peer does not know the full $P_k(t)$ of the linked peer; it instead has a *Temporary Peer Knowledge* (TP_k) which is built based on previously received queries from the source peer. Different meanings of links are modelled by means of a labelling function $Label$: for a given link $l = (s, t) \in L$, $Label(l)$ is a vector of two elements $[e, w]$: the former is the link label and the latter is a weight used to model what the source peer knows of the target peer; this is computed as follow:

- if $e = AL \Rightarrow w = \emptyset$
- if $e = TSL \Rightarrow w = TP_k$
- if $e = FSL \Rightarrow w = P_k(t)$

In the next two sections, we give a brief description of how ***PROSA*** works. A detailed description of ***PROSA*** can be found in [3].

A. Peer Joining to ***PROSA***

The case of a node that wants to join an existing network is similar to the birth of a child. At the beginning of his life a child "knows" just a couple of people (his parents). A new peer which wants to join, just looks for n peers at random and establishes ALs to them. These links are ALs because a new peer doesn't know anything about its neighbours until he doesn't ask them for resources. This behaviour is quite easy to understand: when a baby comes to life he doesn't know anything about his parents. The ***PROSA*** peer joining procedure is represented in algorithm 1.

Algorithm 1 **JOIN:** Peer s joining to ***PROSA***$(\mathcal{P}, \mathcal{L}, P_k, Label)$

Require: ***PROSA***$(\mathcal{P}, \mathcal{L}, P_k, Label), Peer\ s$
1: $\mathcal{RP} \leftarrow rnd(P, n)$ {Randomly selects n peers}
2: $\mathcal{P} \leftarrow \mathcal{P} \cup s$ {Adds s to set of peers}
3: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(s,t), \forall t \in \mathcal{RP}\}$ {Links s with peers in $\mathcal{RP}$}
4: $\forall t \in \mathcal{RP} \Rightarrow Label(s,t) \leftarrow [AL, \emptyset]$ {Labels links as AL}

B. *PROSA dynamics*

In order to show how ***PROSA*** works, we need to define the structure of a query message. Each query message is a quadruple:

$$Q_M = (qid, q, s, n_r) \tag{2}$$

where qid is a unique query identifier to ensure that a peer does not respond to a query more then once; q is the query, expressed according to the used knowledge model[1]; $s \in P$ is the source peer and n_r is the number of required results. ***PROSA*** dynamic behaviour is modelled by Algorithm 2 and is strictly related to queries. When a user of ***PROSA*** asks for a resource on a peer s_q, it builds up a query q and specify a certain number of results he wants to obtain n_r. This is equivalent to call $ExecQuery($***PROSA***$, s_q, n_r)$.

Algorithm 2 **ExecQuery:** query q originating from peer s_q executed on cur

Require: ***PROSA***$(\mathcal{P}, \mathcal{L}, P_k, Label)$
Require: $cur \in \mathcal{P},\ qm \in QueryMessage$
1: $Result \leftarrow \emptyset$
2: **if** $cur \neq s_q$ **then**
3: $UpdateLink($***PROSA***$, cur, s_q, q)$
4: **end if**
5: $(Result, nRes) \leftarrow ResourcesRel($***PROSA***$, q, cur, n_r)$
6: **if** $nRes = 0$ **then**
7: $f \rightarrow SelectForwarder($***PROSA***$, cur, q)$
8: **if** $f \neq \emptyset$ **then**
9: $ExecQuery($***PROSA***$, f, qm)$
10: **end if**
11: **else**
12: $SendMessage(s_q, cur, Result)$
13: $\mathcal{L} \leftarrow \mathcal{L} \cup (s_q, cur)$
14: $Label(s_q, cur) \leftarrow [FSL, P_k(cur)]$
15: **if** $nRes < n_r$ **then**
16: {– Semantic Flooding –}
17: **for all** $t \in Neighborhood(cur)$ **do**
18: $rel \rightarrow PeerRelevance(P_k(t), q)$
19: **if** $rel > Threshold$ **then**
20: $qm \leftarrow (qid, q, s_q, n_r - nRes)$
21: $ExecQuery($***PROSA***$, t, qm)$
22: **end if**
23: **end for**
24: **end if**
25: **end if**

The first time $ExecQuery$ is called, cur is equal to s_q and this avoids the execution of instruction number 3. Next calls of $ExecQuery$, i.e. when a peer receives a query forwarded by another peer, use function $UpdateLink$, which updates the link between current peer cur and peer s_q, if necessary. If s_q is an unknown peer for cur, a new TSL link is added having as weight a Temporary Peer Knowledge(TP_k) based on the received query message. Note that a TP_k can be considered as a "good hint" for the current peer, in order to gain links to other remote peers. It is really probable that the query would be finally answered by some other peer and that the requesting peer will download all resources that matched it. It would be useful to record a link to that peer, just in case that kind of resources would be requested in the future by other peers. If cur has already a TSL link to s_q, the corresponding TPV (Temporary Peer Vector) in the list is updated. If the link from cur to s_q is a FSL, no updates are performed.

The relevance of a query with respect to the resources hosted by cur is evaluated calling function $ResourcesRel$. Two possible cases can hold:

- If none of the hosted resources has a sufficient relevance, the query has to be forwarded to another peer f, called "forwarder". This peer is selected among cur neighbours by the function $SelectForwarder$, using the following procedure:
  - Peer cur computes the relevance between query q and the weight of each links connecting itself to the neighbourhood.
  - It selects the link with the highest relevance, if any, and forward the query message to it.
  - If the peer has neither FSLs nor TSLs, i.e. it has just ALs, the query message is forwarded to one link at random.

  This procedure is described in Algorithm 2, where the subsequent forwards are performed by means of recursive calls to $ExecQuery$.
- If the peer hosts resources with sufficient relevance with respect to q, two sub-cases are possible:
  - The peer has sufficient relevant documents to full-fill the request. In this case a result message is sent to the requesting peer and the query is no more forwarded.
  - The peer has a certain number of relevant documents, but they are not enough to full-fill the request (i.e. they are $< n_r$). In this case a response message is sent to the peer s_q, specifying the number of matching documents. The message query is forwarded to all the links of cur neighbourhood whose relevance with the query is higher than a given threshold (semantic flooding). The number of matched resources is subtracted from the number of total requested documents before each forward step.

When the requesting peer s_q receives a response message it presents the results to the user. If the user decides to download a certain resource from another peer, the requesting peer contacts the peer owning that resource asking for download. If download is accepted, the resource is sent to the requesting peer.

[1]If knowledge is modelled by Vector Space Model, for example, q is a state vector of stemmed terms. If knowledge is modelled by ontologies, q is an ontological query, and so on

IV. TOPOLOGICAL PROPERTIES

Algorithms described in section III are inspired by the way social relationships among people evolve, in the hope that a network based on those simple rules could naturally become a small–world. That of being a small–world is one of the most desirable properties of a P2P network, since resource retrieval in small–worlds is really efficient. This is mainly due to the fact that small–world networks have a short Average Path Length (APL) and a high Clustering Coefficient (CC). APL is defined as the average number of hops required to reach any other node in the network: if APL is small, all nodes of the network can be easily reached in a few steps starting from whichever other node.

CC can be defined in several ways, depending on the kind of "clustering" you are referring to. We used the definition given in [6], where the clustering coefficient of a node is defined as:

$$CC_n = \frac{E_{n,real}}{E_{n,tot}} \tag{3}$$

where n's neighbours are all the peers to which n as linked to, $E_{n,real}$ is the number of edges between n's neighbours and $E_{n,tot}$ is the maximum number of possible edges between n's neighbours. Note that if k is in the neighbourhood of n, the vice-versa is not guaranteed, due to the fact that links are directed. The clustering coefficient of the whole network is defined as:

$$CC = \frac{1}{|V|} \sum_{n \in V} CC_n \tag{4}$$

i.e. the average clustering coefficient over all nodes.

The CC is an estimate of how strongly nodes are connected to each other and to their neighbourhood. In particular, the definition given in Equation 3 measures the percentage of links among a node neighbours with respect to the total possible number of links among them.

In the following two subsections we show that ***PROSA*** has both a small APL and a considerable high CC.

A. Average path length

Since we are focusing on topological properties of a ***PROSA*** network to show that it is a small–world (i.e. that queries in ***PROSA*** are answered in a small amount of steps), we estimate the APL as the average length of the path traversed by a query. It is interesting to compare the APL of ***PROSA*** with the APL of a correspondent random graph, since random graphs usually have a really small average path length.

Given a graph $G(V, E)$ with $|V|$ vertexes (nodes) and $|E|$ edges (links) among nodes, the correspondent random graph is a graph G_{rnd} which has the same number of vertexes (nodes) and the same number of edges (links) of G, and where each link between two nodes exist with a probability p.

Note that the APL of a random graph can be calculated using equation (5), as reported in [14], where $|V|$ is the number of vertexes (nodes) and $|E|$ is the number of edges (links).

$$APL = \frac{\log |V|}{\log \left(|V|/|E|\right)} \tag{5}$$

Figure 1 shows the APL for ***PROSA*** and the correspondent random graph for different number of nodes in the case of 15 performed queries per node. The APL for ***PROSA*** is about 3.0, for all network sizes, while the APL for the correspondent random graph is between 1.75 and 2.0: the average distance among peers in ***PROSA*** seems to be independent from the size of the network. This is quite common in real small–world networks.

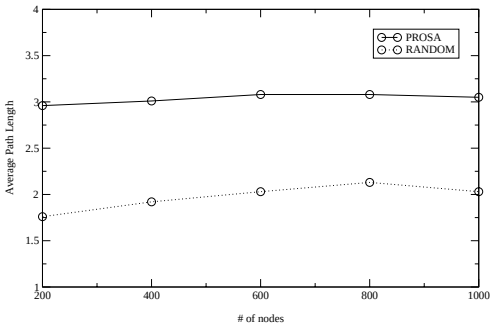

Figure 1. APL for ***PROSA*** and random network

It is also interesting to analyse how APL changes when the total number of performed queries increases. Results are reported in Figure 2, where the APL is calculated for windows of 300 queries, with an overlap of 50 queries. Note that the APL for ***PROSA*** decreases with the number of performed queries. This behaviour heavily depends on the facts that new links among nodes arise whenever a new query is performed (TSLs) or successfully answered (FSLs). The higher the number of performed queries, the higher the probability that a link between two nodes does exist.

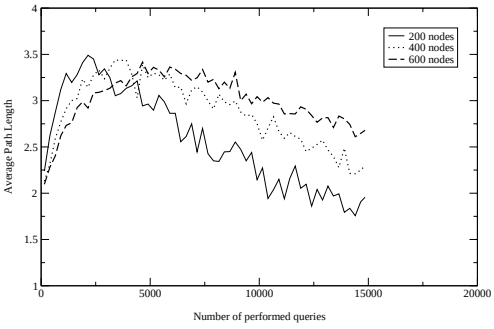

Figure 2. Running averages of APL for ***PROSA*** with different network size

B. Clustering Coefficient

The clustering (or transitivity) of a network is a measure of how strongly nodes are connected to their neighbourhood. Since links among nodes in ***PROSA*** are established as a consequence of query forwarding and

answering, we suppose that peers with similar knowledge will be eventually linked together. This means that usually peers have a neighbourhood of similar peers, and having strong connections with neighbours could really speed–up resource retrieval.

In Figure 3 the CC of ***PROSA*** for different number of performed queries is reported, for a network of 200 nodes. Note that the clustering coefficient of the network increases when more queries are performed. This means that nodes in ***PROSA*** usually forward queries to a small number of other peers so that their aggregation level naturally gets stronger when more queries are issued.

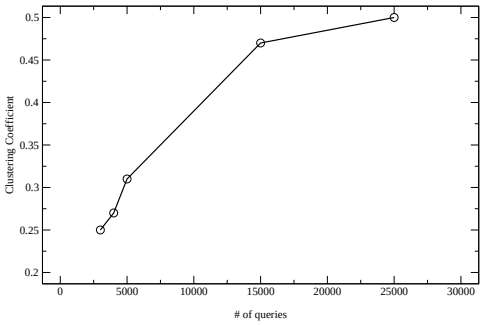

Figure 3. Clustering Coefficient for ***PROSA***

It could be interesting to compare ***PROSA*** clustering coefficient with that of a corresponding random graph. The clustering coefficient of a random graph with $|V|$ vertexes (nodes) and $|E|$ edges (links) can be computed using equation 6.

$$CC_{rnd} = \frac{|E|}{|V| \cdot (|V| - 1)} \tag{6}$$

Figure 4 shows the CC for ***PROSA*** and a correspondent random graph for different network sizes, in the case of 15 performed queries per node. The CC for ***PROSA*** is from 2.5 to 6 times higher that that of a correspondent random graph, in accordance with CC observed in real small–world networks. This result is quite simple to explain, since nodes in ***PROSA*** are linked principally to similar peers, i.e. to peers that share the same kind of resources, while being linked to other peers at random. Due to the linking strategy used in ***PROSA***, it is really probable that neighbours of a peer are also linked together, and this increases the clustering coefficient.

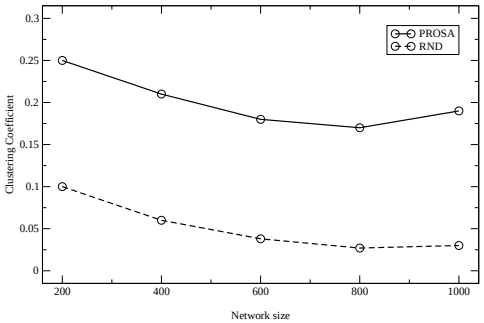

Figure 4. Clustering coefficient for ***PROSA*** and the corr. random graph

V. ***PROSA*** PERFORMANCES

Topological properties reported in the previous section are sufficient to state that ***PROSA*** naturally evolves to a small–world only by means of query forwarding and link updating. This is an important result on his own, since many recent studies are focusing on the problem of building small–world networks with given features.

Nevertheless, since we are mainly interested on using ***PROSA*** for resource sharing, it is of the most importance to investigate its performance in resource searching and retrieval. In this section we report some interesting measures on ***PROSA***, showing how the same algorithms which contribute to obtain a small–world structure also guarantees a fast and effective retrieval of shared resources.

A. Answer Delay

The "answer delay" is defined as the time a user has to wait after a query is issued in order to obtain the first answer. In P2P resource sharing systems, answer delay can be also measured by the so–called "Query deepness".

Query deepness is defined as the length of the path traversed by a query during searching; it is strictly related with the existing time interval between the time when the query is issued and the time when answers to the query are obtained. If queries do ideally require almost the same amount of time to be processed by a node, the higher the query deepness is, the longer an user has to wait to obtain an answer.

In P2P resource sharing applications a low query deepness is a desirable feature, since it heavily impacts on the quality of service observed by users: obtaining all resources matching a given query only after twenty or thirty seconds is worst than obtaining the first twenty or thirty percent of results in a couple of seconds, waiting a bit longer for the remaining results.

Since paper about SETS and GES do not report any values about query deepness, figure 5 reports the average query deepness measured for different network sizes for ***PROSA*** and a random walk strategy. Reported measures for each network size are performed on the same network.

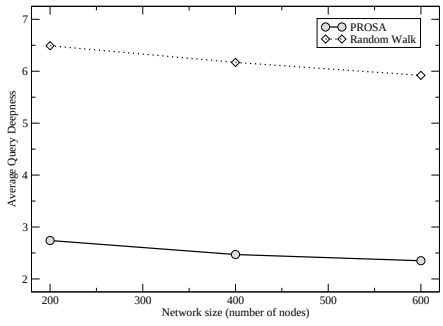

Figure 5. Average query deepness

As showed in figure 5, the average query deepness for ***PROSA*** is about one half that of a random walk, for all network sizes. This means that nodes in ***PROSA*** are able

to find the most convenient nodes to forward a query to, sensibly lowering the average delays of answers: random walk is not the best strategy for query forwarding in a non–random network like ***PROSA*** is. Note also that the global effect of a low query deepness measured on ***PROSA*** is obtained through simple local choices made by each node, looking only at properties of nodes in his neighbourhood.

B. Query Recall

One of the most used quality measure for P2P retrieval system is the "recall", defined as the average percentage of resources retrieved by each query over the total amount of matching resources available in the network. So if a query matches N resources in the network, and it is eventually returned K resources, with $K \leq N$, the obtained recall is $\frac{K}{N}$.

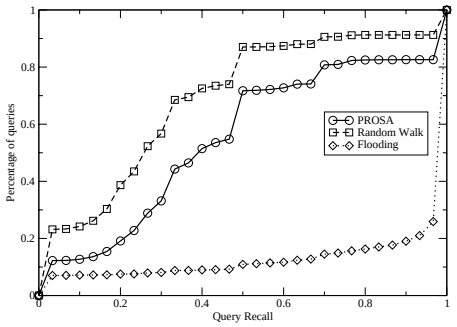

Figure 6. Cumulative recall distribution over queries

The cumulative distribution of recall over queries for ***PROSA*** is reported in figure 6 and compared with the recall distribution of flooding and random walk measured on the same network. More than 50% of queries in ***PROSA*** have a recall higher than 50%, while the random walk is able to obtain 50% of recall for less than 10% of nodes. If we consider that the average processing cost for ***PROSA*** is ever lesser than in random walk, we can conclude that ***PROSA*** is a great improvement over pure random walk for resource discovery.

C. Processing Cost

Another interesting quality measure for a routing strategy is the so–called "Processing Cost", defined as the average number of nodes involved in answering each query. It is an approximated measure of the average computational power needed to answer a query, and is related to network scalability. Having a high Processing Cost implies that each query is forwarded to a considerable percentage of nodes, consuming a high amount of bandwidth and computational resources. The processing cost for flooding, for example, is of 100% in a connected network, since queries are forwarded to all outgoing links recursively: everybody knows that flooding is not the optimal solution for bandwidth usage, while being the optimal one for recall.

Algorithms for resource searching usually have a different processing cost for different values of desired recall. In GNUTELLA-like systems, for example, the trade–off between processing cost and recall is managed by fixing the maximum deepness of a query message, at the price of a possibly lesser recall, since nodes beyond the prefixed deepness could actually have relevant documents, but are not reachable at all.

In both SETS and GES approaches, if a query did not gather a sufficient amount of answers it is forwarded along other routing paths, and a flooding–like effect is obtained. The case of ***PROSA*** is a little different. Queries are possibly forwarded to relevant nodes until the first matching resource is obtained, and then to all nodes that are really relevant to the query, till it is possible.

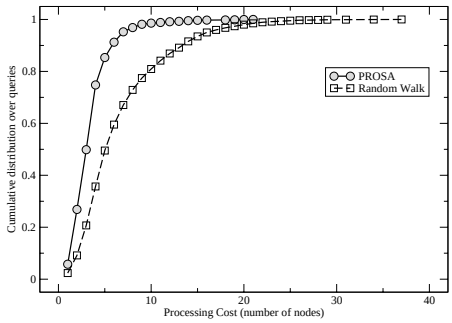

Figure 7. Cumulative distribution of processing cost over queries

Figure 7 reports the cumulative distribution of processing cost over successful queries for ***PROSA*** and a random walk strategy for a network with 200 nodes. In this graph a query is considered "successful" if it obtained at least one matching resource. Note that more than 90% or queries are answered in ***PROSA*** by visiting less than 4% of the nodes, while a random walk needs to probe at least 17% of nodes to answer the same percentage of queries.

These results about processing cost, aside with low query deepness discussed above, confirm that the routing strategy implemented by ***PROSA*** is powerful and efficient: the majority of queries find matching resources just two or three hops away, and 90% of them are answered with at least one resource by probing less than 4% of the nodes. These features let us assert that scalability would not be a hard issue for ***PROSA***. Finally, the processing cost needed to obtain at least one matching resource is many times lower in ***PROSA*** than in SETS and GES, As showed in [2] and [1], both systems are able to obtain a recall higher than 90% with a processing cost not lesser than 50%. This means that half of the network should be probed in order to obtain a relevant amount of matching resources, and this would be practically unfeasible in very large networks.

VI. CONCLUSIONS AND FUTURE WORK

PROSA is a P2P system mainly inspired by social networks and behaviours. Topological properties of ***PROSA*** suggest that it naturally evolves to a small–world network,

with a very short average path length and a high clustering coefficient. This ***PROSA*** emerging behaviour is exploited in order to implement a fast and efficient query routing algorithm. Results about query efficiency are confirmed by simulation results reported. Moreover, in this paper a comparison of ***PROSA*** with other recent algorithms for P2P semantic resources organisation (such as SETS [1] and GES [2]) is presented.

Future work includes deeply examining the internal structure of ***PROSA*** networks and studying the emergence of communities of similar peers.

REFERENCES

[1] M. Bawa, G. S. Manku, and P. Raghavan, "Sets: search enhanced by topic segmentation," in *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. New York, NY, USA: ACM Press, 2003, pp. 306–313.

[2] Y. Zhu, X. Yang, and Y. Hu, "Making search efficient on gnutella-like p2p systems," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE Computer Society, April 2005, pp. 56a– 56a.

[3] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia, "Social behaviours applied to p2p systems: an efficent algorithm for resources organisation," 2006.

[4] ——, "Exploiting social networks dynamics for p2p resource organisation," in *Computer and Information Sciences – ISCIS 2006*, ser. LNCS, no. 4263, 2006, pp. 726–734.

[5] M. Buchanan, *Nexus: Small Worlds and the Groundbreaking Theory of Networks*, 2003.

[6] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[7] S. J., *Social Networks Analysis: A Handbook*. Sage Pubblications, London, 2000.

[8] M. E. J. Newman, "The structure of scientific collaboration networks," *PROC.NATL.ACAD.SCI.USA*, vol. 98, p. 404, 2001. [Online]. Available: http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0007214

[9] R. J. Williams, E. L. Berlow, J. A. Dunne, A.-L. Barabsi, and N. D. Martinez, "Two degrees of separation in complex food webs," *Proceedings of the Nat'l Academy of Sciences*, vol. 99, pp. 12 913–12 916, 2002.

[10] R. Albert and A.-L. Barabasi, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, p. 47, 2002. [Online]. Available: http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0106096

[11] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, ser. Lecture Notes on Computer Science. Springer Verlag, 2005, vol. 3485.

[12] "Gnutella website." World Wide Web, http://www.gnutella.com.

[13] H. Schutze and C. Silverstein, "A comparison of projections for efficient document clustering," in *Prooceedings of ACM SIGIR*, Philadelphia, PA, July 1997, pp. 74–81.

[14] M. E. J. Newman, "Scentific collaboration networks. ii. shortest paths, weighted networks, and centrality," *Physical Review E*, vol. 64, p. 016132, 2001. [Online]. Available: http://www-personal.umich.edu/ mejn/papers/016132.pdf

Vincenza Carchiolo is full professor of Computer Science in Department of Informatics and Telecommunications at University of Catania. Her research interests include information retrieval, query languages, distributed system, and formal language. She received a degree with Honours in Electrical Engineering from University of Catania, Italy in 1983. She is member of ACM.

Michele Malgeri is associate professor in Department of Informatics and Telecommunications at University of Catania. His research interests include distributed system, information retrieval, query languages and formal language. He received a degree with Honours in Electrical Engineering from University of Catania, Italy in 1983.

Giuseppe Mangioni received the degree in Information Engineering (1995) and the Ph.D. degree (2000) at the University of Catania (Faculty of Engineering), where he became a professional engineer in 1995. In 1996 he joined the Dept. of Information and Telecommunications Engineering as a contract researcher. Currently he is a contract professor in Computer Networks at the Faculty of Engineering of Catania. Presently, his main research interests concern e-learning, P2P overlay networks and complex systems.

Vincenzo Nicosia received his degree in Computing Engineering (2004) and is currently Ph.D. student at Dept. of Informatics and Telecommunications Engineering. His research interests include distributed systems, security, peer–to–peer networks and complex systems.