To appear in European Performance Engineering Workshop, EPEW'07

Evaluation of P2P Search Algorithms for Discovering Trust Paths

Emerson Ribeiro de Mello[⋆1], Aad van Moorsel[⋆⋆2], and Joni da Silva Fraga[1]

[1] Department of Automation and Systems
Federal University of Santa Catarina
Florianópolis, SC - Brazil
emerson@das.ufsc.br,fraga@das.ufsc.br
[2] School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
aad.vanmoorsel@newcastle.ac.uk

Abstract. Distributed security models based on a 'web of trust' eliminate single points of failure and alleviate performance bottlenecks. However, such distributed approaches rely on the ability to find trust paths between participants, which introduces performance overhead. It is therefore of importance to develop trust path discovery algorithms that minimize such overhead. Since peer-to-peer (P2P) networks share various characteristics with the web of trust, P2P search algorithms can potentially be exploited to find trust paths. In this paper we systematically evaluate the application of P2P search algorithms to the trust path discovery problem. We consider the number of iterations required (as expressed by the TTL parameter) as well as the messaging overhead, for discovery of single as well as multiple trust paths. Since trust path discovery does not allow for resource replication (usual in P2P applications), we observe that trust path discovery is very sensitive to parameter choices in selective forwarding algorithms (such as K-walker), but is relatively fast when the underlying network topology is scale-free.

Keywords: Peer-to-Peer, Web of Trust, Trust Paths

⋆ Supported by CNPq. This work was conducted while the first author was in the School of Computing Science at Newcastle University, UK. This work has been developed within the scope of the "Security Infrastructure for Service Oriented Distributed Applications" project (CNPq 550114/2005-0).

⋆⋆ Supported in part by EPSRC grant EP/C009797/1 "Dynamic Operating Policies for Commercial Hosting Environments" and EU Network of Excellence grant 026764 "Resilience for Survivability in IST".

1 Introduction

The effectiveness and efficiency of any commercial interaction depends strongly on the level of trust that exists between involved parties. Trust determines if parties are willing to depend on each other, even if negative consequences are possible [1], and without it, commercial transactions will be inefficient because of doubts about payments, ability to deliver a service, etc. Trust establishment in real life is usually a complex and subjective process, and in electronic commerce, trust establishment arguably becomes even more challenging [1]. The absence of human interaction and the frequency and speed with which new electronic commerce interactions can be established contribute to this challenge.

Several automated trust solutions have been proposed in the literature and some are in common use, such as, X.509 [2], PGP [3] and SPKI/SDSI [4,5]. These solutions provide ways of determining and assuring that information is being exchanged with a trusted source. Of particular interest in large scale deployment of trust solutions is the notion of a *web of trust*. In a web of trust each party has the ability to express their trust in entities and communicate this to other parties (by signing messages). By association, these other parties may decide to trust the entities as well. Web of trust solutions are in contrast to the traditional model that relies on the trust in central entities, namely the Certification Authorities (CA).

The literature discusses trust relation creation and management extensively [6,7,8,9], and the PGP and SPKI/SDSI standard proposals discuss the concept of trust paths as well. Recently, various authors have proposed algorithms for discovering trust paths [10,11,12] to fill the void left by the standards (which purposely do not specify how trust paths should be discovered), but no experimental or simulation results have been presented to study the effectiveness and performance of the algorithms.

Trust path discovery is equivalent to finding paths in graphs (we make this precise in Section 2). To be of practical value, a trust path discovery algorithm must take into account that participants are only aware of their direct neighbours. This immediately suggests that P2P search algorithms may be applicable to this domain, as also realized in [10,11,12]. After all, in P2P networks each node keeps track of a partial index with a subset of all nodes of the network. In this paper we therefore evaluate how existing P2P algorithms perform when applied to the trust path discovery problem.

There exist important differences between traditional P2P applications and the web of trust. In particular, in a typical P2P application (such as file sharing), files will be replicated across peers, thus allowing for tremendous scaling. In the web of trust, however, a trust relationship will be present only in the two nodes that compose the trust relationship. Replication is possible in some settings (as we will explain in Section 2), but will be far less prevalent and straightforward than in traditional P2P applications. As a consequence, our simulation results will show that an underlying scale-free network topology performs relatively well for the trust path discovery problem compared to resource discovery in traditional P2P applications (as reported in [13,14,15]). Furthermore, we will see

that the performance of the search algorithms is extremely sensitive to parameter choices in modified flooding algorithms that limit the amount of forwarding (such as selective querying and K-walker).

2 Trust Path Discovery Problem

In abstract terms, the web of trust can be seen as a graph, where the nodes are participants and the arcs denote trust (an arc from A to B denotes that A trusts B). In terms of PGP, one can restate this as nodes being keys and arcs being signatures that signify trust, e.g., [16]. Arcs can be uni-directional or bi-directional, since the trust relationship could be one-way or two-way. For instance, in the X.509 model the users trust in the Certificate Authorities but the inverse is not true. Thus, in this case we have a one-way trust relationship. In PGP and SPKI model each principal can be the issuer or the subject of a trust relationship, amounting to a two-way trust relationship. Such two-way relations can be implemented through various mechanisms, for example through exchange of two signed certificates. In this paper we assume the PGP and SPKI model, in which trust relations are bi-directional, i.e., the underlying graph is undirected.

In a web of trust, if there is no trust relation between two parties A and C, they can still trust each other if there exists at least one path between A and C in the graph (we will call A the origin, and C the target in what follows). That is, we exploit the fact that trust can be said to be transitive [17,18]: if A trusts B, and B trusts C, then A trusts C. The trust path discovery problem then is to find at least one trust path between two given principals. Discovering such a path is complicated because each node has only knowledge about its own trust relations. This suggests, however, that unstructured P2P search algorithms are natural candidates to solve the trust path discovery problem.

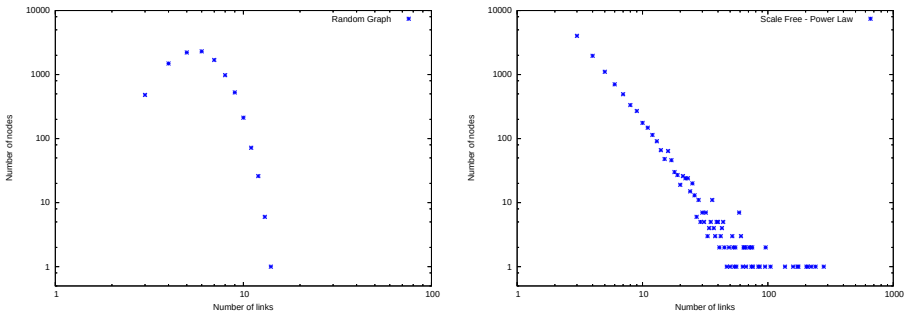
2.1 P2P networks

There are two categories of decentralized P2P networks: *unstructured*, such as Gnutella [19], and *structured*, such as those based on a distributed hash table [20,21]. We will discuss trust path discovery only for unstructured approach, since we found that the mapping of a web of trust on a structured P2P network does not seem to provide any benefits for discovering trust paths. In traditional P2P networks, one would search for 'resources'. When searching trust paths the 'resources' a node contains are the trust relationships it knows about. In the default setting, each node only knows about its own trust relationships, and thus in the underlying P2P network each node is connected to the nodes it trusts. Discovering a trust path between A and B then is identical to A querying for a resource that is only present at B.

To find resources (such as files) in unstructured P2P networks, each query is propagated through the network by *flooding*. For instance, in the original version of Gnutella [19] a node receiving or generating a query forwards it to a fixed number of neighbours (typically four). These neighbours forward it to

their neighbours, and so on until the message time to live (TTL) threshold (typically seven) has been exceeded. Flooding can directly be applied to the problem of discovering trust paths. If existing, trust paths will be discovered using exhaustive flooding.

Flooding has an obvious disadvantage, namely that many query messages may be needed to find a resource. When establishing trust paths, this problem is magnified by the fact that there is no replication of resources (i.e., trust relationships) across multiple nodes. Several variations and modifications of straightforward flooding have been proposed for traditional P2P networks. Depth-first search was used in [13,14,22], and breadth-first search with incrementing message time to live values was proposed and analysed in [15,23,24]. In the Kazaa network [25] the concept of super-nodes (or ultra-peers) was introduced to create a hierarchical structure in the network, where queries are propagated on a super-nodes overlay that acts like shortcuts between distant principals. An improvement important for the problem of trust path discovery is that of caching 'hits'. In these solutions a "queryHit" message is created when a desired trust path is found, and this message is propagated in the reverse path. Subsequent queries then can immediately use the cached result. This approach is very beneficial for discovering trust paths, and we will evaluate it below.



(a) random topology (b) scale-free topology

Fig. 1. Distribution of trust relations ('links') per node in different network topologies.

3 Experiment Setup

3.1 Trust Path Discovery Algorithms

We will compare the following trust path discovery algorithms, which all are variations of flooding in unstructured P2P networks. For the evaluation of these approaches in the context of file sharing and similar applications, we refer to, e.g., [13,14,15,22,23].

K-walker. In K-walker every node propagates the query to K randomly selected neighbour nodes, resulting in a variation of breadth-first search.

Selective querying. Selective querying works like K-walker, but a query will be propagated through the K *best* nodes according to some specific criterion, for instance, location, bandwidth, number of neighbours, results in previous queries, etc. In trust path discovery, it seems logical to select nodes with the most neighbours, since this implies these nodes contain more trust relationships.

Expanding ring. The expanding ring approach repeatedly increments the message time to live (TTL) until the trust path is found. Initially the search starts with a small value of TTL (expressed in terms of the number of hops) and if the search does not succeed the TTL value will be increased and the same query will be sent. This process repeats itself until some pre-defined maximum value for TTL is reached. The reasoning behind the expanding ring approach is that it avoids the problem that if a resource is found but TTL has not been reached, unnecessary messages continue to be forwarded.

Ultra-peers. The ultra-peer approach introduces a hierarchy between nodes. When a leaf node joins the network it associates itself with one or more ultra-peers (super-nodes). In the context of trust paths, each ultra-peer stores information about trust relationships of its leaf nodes. The search started by a leaf node will be propagated only in the ultra-peer layer, since these nodes already know about the trust relations the leaf nodes provide. It is important to note that introducing a node hierarchy is against the philosophy of the web of trust, in which all nodes are equal. However, one can imagine cases in which ultra-peers naturally arise (for instance in the shape of CAs), and we therefore study the ultra-peer performance and efficiency as well.

Cache table. In the cache table approach, each node in the network has a cache table that stores references about previous successful searches. If the same target is used again, the path will be found faster. The cache table can be used with any of the above algorithms, but in our experiments we only study it in combination with Gnutella-like flooding.

3.2 Topologies

The topology of a P2P network heavily influences the effectiveness of various algorithms. The topology is determined by the number of neighbours each node knows about (the degree of a node), and in our study we consider two classes of topologies: the random graph and the scale free or power law graph [26].

There exist different variations of random graphs and various approaches to generating random graphs. We use one of the standard approaches provided by Peersim (see below), namely one that generates for each node a fixed number d of edges, and then connects these edges with d randomly selected neighbouring nodes. Since our simulations use undirected graphs, this results in an average degree of $2d$ for each node. Figure 1(a) shows (using logarithmic scale) the number of nodes with a certain degree, where the average degree for a node is 4. In the

simulation, we generated random graphs with up to 20,000 nodes, and average degree 4.

Scale free graphs follow the power law distribution where many nodes have few connections and few nodes have many connections. This kind of distribution represents the small world concept [27] observed in several different areas, including in the web of trust [28] and traditional P2P file-sharing applications [29]. We use the Barabasi-Albert approach [26] to generate scale free graphs (using the implementation provided in Peersim, see below). Again, we set the average degree to 4. An example of the degree of nodes in the resulting graph is given in Figure 1(b).

Peersim - A P2P Simulator. We used the P2P network simulator Peersim [30] to carry out the discrete-event simulations. Peersim is implemented in Java; it generates networks according to several possible topology classes (including random and scale-free topologies) and has a discrete-event simulator. The simulation execution scales very well when using Peersim's 'cycle-based' approach, which ignores certain transport layer elements and concurrency, as recommended in [31]. Peersim also provide a way to create independent replicas of experiments based on a pseudo random generator, which we used to gain confidence in our simulation results. Using Peersim's Java API, various P2P algorithms can be very quickly implemented and evaluated. The simulation runs for which we present results in the next section typically lasted on the order of (tens of) seconds.

4 Results

There are a number of metrics one may want to consider when establishing the quality of a trust path discovery algorithm. In this section, we will first discuss the cost of establishing a single trust path, and then multiple trust paths. (The latter may be important because the existence of multiple trust paths may increase the trust level the origin associates with the target.) The cost we consider is the number of messages passed over the network to find the trust path(s). We will also analyse the sensitivity of the results with respect to the TTL value, which is a critical parameter that needs to be set in all algorithms. Moreover, in absence of a notion of time in our simulation, the minimum required TTL may also be used as an indicator of the time it will take to find the trust paths. As we mentioned above, the simulation is based on graphs with 20,000 nodes for both the random and scale free graph topology, with average node degree 4. We ran simulations with three different distances from the (arbitrary chosen) origin node: closest (that is, two hops), average and farthest node from the origin node. For the random topology, the distances were as follows: average is 7 hops and farthest is 10 hops; for the scale free topology: average is 4 hops and farthest is 6 hops. In our simulations, for all algorithms, TTL was incremented from 2 until 7, and sometimes increased higher to observe specific phenomena. For the algorithms selective querying and K-walker we chose three different values for the number of neighbours to which the query will be

propagated: 10%, 50% and 70%. In the ultra-peers algorithm the number of super-nodes in the network was chosen randomly, selecting the most connected nodes. The amount of nodes selected to be ultra peers was chose by the integer part of the square root of network size.

TTL	Gnutella		K-walker			Selective			Ultra-peers
	original	cacheTable	10%	50%	70%	10%	50%	70%	
5	0	0	0	0	0	0	0	0	0
6	2	2	0	0	2	0	0	1	2
7	3	3	0	0	2	0	0	1	3
10							first hit		
11				first hit					
32			first hit						

Table 1. Number of trust paths found for random graph topology for different TTL values. Target is 7 hops away. The last three lines indicate required TTL value to find at least one path but even with big TTL values "Selective 10%" did not find any path.

Discovery of the first trust path. Table 1 and Table 2 show the number of trust paths found, for different values of TTL, for the random and scale-free topology, respectively. The graphs in Figure 2 and Figure 3 show the number of messages propagated through the network for each algorithm, for different values of TTL. We can see in Figure 2 and Figure 3 that both the network topology and the specific algorithm have an important influence on the number of messages propagated. In scale free networks some nodes have a high number of neighbours, resulting in more messages in the network when flooding techniques are used. Moreover, these messages can be redundant, because they forward a query to a node that earlier received that same query (see [14] for an in-depth discussion). However, Table 1 and Table 2 demonstrate the benefit of this higher number of messages in the scale-free topology: the trust path is found within only a few hops for all algorithms. More precisely, the trust path is found in the minimum number of hops (TTL=3 for a target 4 hops away), except for K-walker with 10% forwarding. In the random graph, trust paths are not always found for low values of TTL. The minimum possible value of the TTL is 6 for a target at the average distance of 7 hops, but algorithms that do not forward to a high number of neighbours require higher TTL values. Table 1 shows this number in the three last rows: to find the first trust path K-walker with 50% requires a TTL value of 11, and K-walker wit 10% needs TTL value 32. Moreover, selective forwarding with 10% never reaches the target, as indicated in Table 1.

If one is interested in the number of messages used to find the first trust path, one combines the above-mentioned figures and tables. For the scale free topology, the Selective and K-walker algorithms with only 10% forwarding work best, using only 178 and 425 messages, respectively. For comparison, the original flooding (Gnutella) method would generate close to 4000 messages before it finds the first

TTL	Gnutella		K-walker			Selective			Ultra-peers
	original	cache	10%	50%	70%	10%	50%	70%	
2	0	0	0	0	0	0	0	0	1
3	1	2	0	1	1	1	1	1	3
4	4	19	0	3	2	1	2	3	7
5	4	95	0	3	3	1	2	3	9
6	5	138	1	4	4	1	3	3	10
7	5	180	1	5	5	1	3	3	11

Table 2. Number of trust paths found for scale free topology for different TTL values. Target is 4 hops away.

path. For the random graph, Selective and K-walker still outperform flooding, but only if the forwarding is at a high level (70%). To illustrate this, flooding uses about 2700 message to find the first trust path, and K-walker with 10% and 50% require as many or more: 2200 and 9000, respectively (the latter numbers are not visible in Figure 2 because they require higher values of TTL than displayed there). However, K-walker with 70% and Selective with 50 and 70% require less than 1000 messages. In other words, we notice extreme sensitivity to the chosen amount of forwarding in the K-walker as well as Selective querying algorithm.

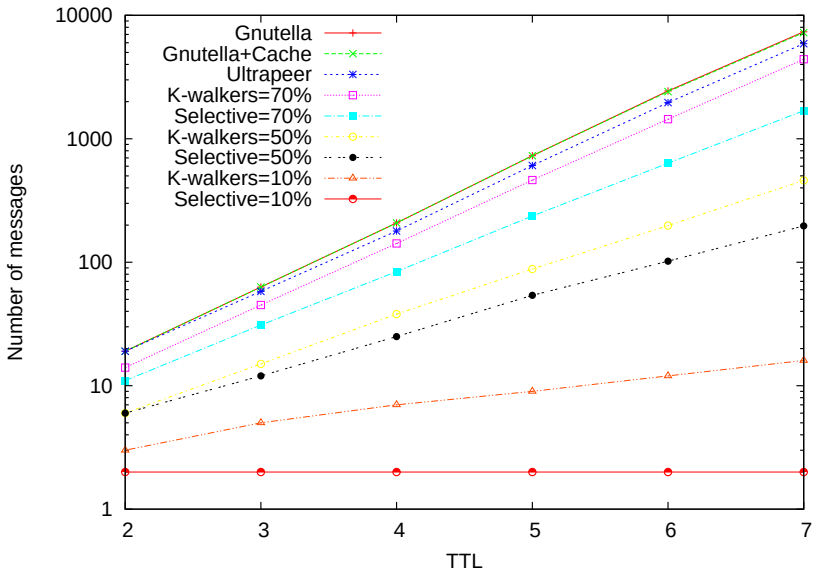


Fig. 2. Number of messages under different TTL values: Random graph topology – Distance = 7 hops

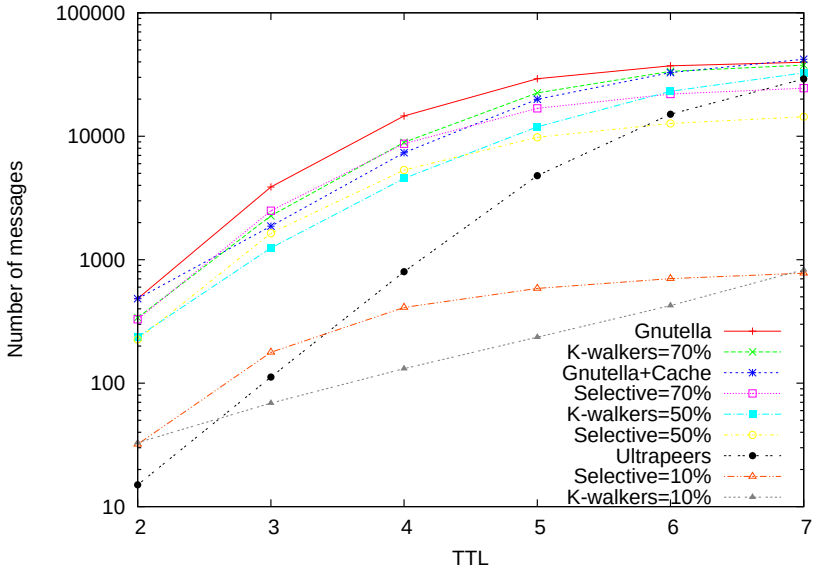


Fig. 3. Number of messages under different TTL values: Scale free topology – Distance = 4 hops

If we compare the above findings with for instance [14], where networks of similar size were simulated, we note that the main difference is the lack of replication of the target in our setting. Hence, the conclusion drawn in [14] that scale-free topologies should be avoided does not necessarily hold for discovering trust paths. We find that the amount of messages needed to find the first trust path is similar for both topologies, and is very sensitive to specific parameter choices such as the K value in K-walker. However, trust paths can be found for low values of TTL in scale-free topologies–this indicates that the time it takes to discover a trust path will be less for the scale-free topology, and this also indicates that the expanding ring approach will work well for the scale-free topology.

Performance of individual algorithms. The distributed flooding algorithm (labelled 'Gnutella' in figures and tables) is known to be expensive in terms of the amount of messages used, and certainly for the scale-free topology our results confirm this insight. Instead, it is better to limit the number of forwarded messages using selective or K-walker with values as low as 10% (as long as the topology is scale-free). Flooding with caching ('Gnutella + Cache') did not differ much from Gnutella, except in terms of the number of paths found, a result we discuss in more detail below. We obtained the results for the caching algorithm as follows. We conducted several consecutives searches using the same origin and target nodes. The values obtained with the first search were cached. In the second search each node that has the trust path to the target in its cache

will reply with this answer. The same query was repeated until every neighbour of the source node obtained cached results. In our test environment we needed four consecutive queries, at which moment we obtained the results presented in the tables and figures.

As we mentioned above, selective querying or K-walker algorithms are natural alternatives for flooding, typically outperforming it. However, selective querying proves a more stable solution, in that K-walker can lead to very poor and hard to predict results in terms of the number of messages needed. To find one trust path with $K = 50\%$ in the random graph topology leads to a worst case scenario (9000 messages, quadruple the number needed by Gnutella). In our simulation, selective querying forwards to the nodes with the higher number of neighbours (as opposed to the randomly selected neighbours in K-walker), and that pays off in the random graph topology as well.

The ultra-peers algorithm in the random graph topology uses an amount of message very similar to the Gnutella protocol. However, in scale free graphs this algorithm got superior results, with trust paths found in as little as 15 messages. The reason for this is that we select the nodes with the highest number of neighbours as ultra peers, a fact that hardly helps if the topology is random, but works very well in the scale-free case.

In a practical implementation one needs a mechanism to increase the TTL value using expanding rings (or use one of the more advanced suggestions in [14]). The results for expanding ring are the sum of the results for individual TTL values. Therefore, it is important that an algorithm finds the trust path for a low value of TTL. In that respect, the results for the scale-free topology compare favourably to those for the random topology. So, even though selective forwarding to 50% of the neighbours in the random topology, with TTL equal to 32 may require as little as 200 messages, following the expanding ring approach the search would have to be repeated for increasing TTL values, thus proving costly after all. The scale-free topology, on the other hand, requires only few iterations in the expanding ring approach because the TTL needed is low.

Discovery of multiple trust paths. One can argue that if multiple trust paths to the target are known, one can place more trust in the target. How to quantify trust as a function of the number of paths is beyond the scope of this paper, but it is of interest to study the performance of the various algorithms when multiple paths should be found. Figure 4 shows for the scale-free topology the number of messages needed to discover multiple paths. We see that the algorithms differ little except for the cached and ultra-peer variants. It seems that when multiple paths need to be found, we come closer to a situation of exhaustive search throughout the network, at which time the chosen algorithm becomes unimportant. It therefore seems inevitable that hierarchical or caching solutions are implemented if the objective is to find more than one trust path to targets.

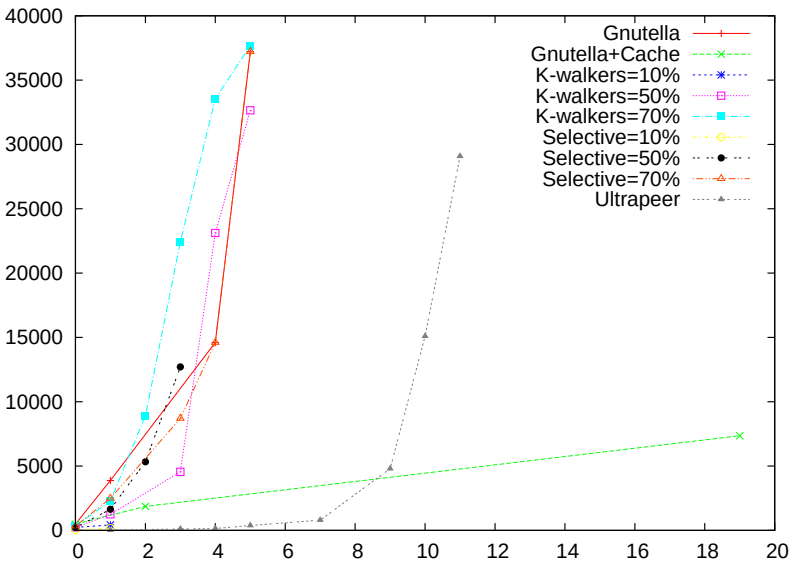


Fig. 4. Number of messages for different number of discovered trust paths: Scale-free graph topology – Distance = 4 hops

5 Conclusions

P2P search algorithms are obvious candidates for discovering trust paths in distributed versions of the web of trust, and we therefore present in this paper a performance comparison of P2P search algorithms when applied to the trust path discovery problem. From our experiments we conclude that the algorithms perform relatively well for a scale-free network topology, especially when compared to traditional file sharing applications, and we argued that the reason for this is that replication of resources has no counterpart in trust path discovery. We also saw that in trust path discovery the performance of restrictive forwarding algorithms such as K-walker can be extremely sensitive to the value of K, the amount of nodes to which a query is forwarded. Furthermore, we obtained results that indicate that when one is interested in discovering multiple trust paths, most algorithms become prohibitively expensive and instead ultra-peer or caching alternatives need to be explored.

References

1. Patil, V., Shyamasundar, R.: Trust management for e-transactions. Sadhana **30**(2 and 3) (April 2005) 141–158
2. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280. (April 2002)

3. Zimmerman, P.: PGP User's Guide. Massachusetts Institute of Technology (May 1994)
4. Ellison, C.M., Frantz, B., Lampson, B., Rivest, R., Thomas, B.M., Ylonen, T.: SPKI Certificate Theory. Internet Engineering Task Force RFC 2693. (September 1999)
5. Rivest, R.L., Lampson, B.: SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession (1996)
6. Jøsang, A., Keser, C., Dimitrakos, T.: Can we manage trust? In: 3rd International Conference on Trust Management – iTrust. (2005) 93–107
7. Skogsrud, H., Benatallah, B., Casati, F.: Model-driven trust negotiation for web services. In: IEEE Internet Computing, IEEE Computer Society (December 2003) 45–52
8. Spantzel, A.B., Squicciarini, A.C., Bertino, E.: Integrating federated identity management and trust negotiation. Technical Report 2005-46, CERIAS – Purdue University (2005)
9. Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., Yu, L.: Negotiating trust on the web. In: IEEE Internet Computing. Number 6 in 6, IEEE Computer Society (December 2002) 30–37
10. Atif, Y.: Building trust in E-commerce. IEEE Internet Computing **6**(1) (2002) 18–24
11. de Mello, E.R., da Silva Fraga, J., Santin, A.O.: O uso do spki/sdsi em redes p2p. In: I Workshop sobre Redes Peer-to-Peer (WP2P'05), Fortaleza, CE - Brasil, XXIII Simpósio Brasileiro de Redes de Computadores (SBRC'05) (2005)
12. Santin, A.O., da Silva Fraga, J., Siqueira, F., de Mello, E.R.: Federation web: A scheme to compound authorization chains on large-scale distributed systems. In: 22nd Symposium on Reliable Distributed Systems (SRDS'03), Florence - Italy (2003)
13. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: INFOCOM. (2004)
14. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. Proceedings of the 16th international conference on Supercomputing (2002) 84–95
15. Zhuang, Z., Liu, Y., Xiao, L., Ni, L.M.: Hybrid periodical flooding in unstructured peer-to-peer networks. In: ICPP, IEEE Computer Society (2003) 171–178
16. Penning, H.P.: Analysis of the strong set in the pgp web of trust (2006) `http://www.cs.uu.nl/people/henkp/henkp/pgp/pathfinder/plot/`.
17. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Trans. on Computer Sys. **8**(1) (February 1990) 18
18. Jøsang, A., Pope, S.: Semantic Constraints for Trust Transitivity. Volume 43 of Conferences in Research and Practice in Information Technology. ACS, Newcastle, Australia (2005)
19. Gnutella: The Gnutella Protocol Specification v0.4. Clip2. (2001)
20. Rowstron, A., Druschel, P.: Pastry: scalable, decentraized object location and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). (November 2001)
21. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw **11**(1) (2003) 17–32

22. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making gnutella-like P2P systems scalable. In: Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM (August 2003) 407–418
23. Jiang, H., Jin, S.: Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. In: ICNP, IEEE Computer Society (2005) 122–131
24. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. In: ICDCS. (2002) 5–14
25. : Kazaa media desktop. `http://www.kazaa.com` (2001)
26. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics **74** (2002) 47
27. Milgram, S.: The small world problem. Psychology Today **1** (1967) 61
28. Capkun, S., Buttyan, L., Hubaux, J.P.: Small worlds in security systems: an analysis of the PGP certificate graph. In: Proceedings of the 2002 New Security Paradigms Workshop. (September 2002) 28–35
29. : Gnumap project. `http://home.comcast.net/~gregory.bray` (2002)
30. : Peersim p2p simulator (2004) `http://peersim.sourceforge.net`.
31. Jesi, G.P.: Peersim howto: Build a new protocol for the peersim 1.0 simulator. (December 2005)