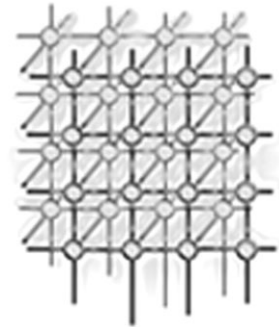

Trust-based probabilistic search with the view model of peer-to-peer networks



Hai Zhuge^{1,*,\dagger}, Xue Chen^{1,2} and Xiaoping Sun^{1,2}

¹China Knowledge Grid Research Group,

Key Lab of Intelligent Information Processing, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing 100080, People's Republic of China

²Graduate School of the Chinese Academy of Sciences, Beijing 100080,
People's Republic of China

SUMMARY

The fundamental challenge confronting peer-to-peer (P2P) resource-sharing networks is to provide efficient and scalable search services in a large-scale, open and dynamic environment. However, current search mechanisms are not efficient or scalable enough with the expansion of the systems. To improve search efficiency, this paper proposes the trust-based probabilistic search mechanism, called the Preferential Walk (*P-Walk*). Every peer adjusts its neighbors' rank continuously according to their performance in previous searches. The basic principle of the *P-Walk* is that neighbors with higher ranks in the question domain have higher probabilities of being queried. Extensive simulation results show that the *P-Walk* can significantly improve the success rate of the query and the response time whilst also effectively reducing traffic volume and controlling malicious behavior. Furthermore, we measure the rank distributions of the peers and find their power-law-like trends based on a Gnutella-like network. Implications are drawn from the analysis of the rank distributions. The *P-Walk* mechanism works effectively in a P2P view model that provides appropriate views of the P2P network for queries of different types. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: P2P; search; probability; trust; power-law; view

1. INTRODUCTION

The advent of peer-to-peer (P2P) networks gradually changed the way we use the Internet and other networks. The term 'peer' indicates that all nodes in the network are treated equally. The main concept

*Correspondence to: Hai Zhuge, China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, People's Republic of China.

^{\dagger}E-mail: zhuge@ict.ac.cn

Contract/grant sponsor: National Basic Research Program of China; contract/grant number: 2003CB317000

Contract/grant sponsor: National Science Foundation of China; contract/grant numbers: 60273020 and 70271007



behind the P2P is decentralization. Peers are able to directly communicate with one another without centralized servers to achieve high efficiency and flexibility in resource sharing and retrieval.

The existing P2P networks can be roughly categorized into two types: unstructured and structured P2P networks. Gnutella [1], Napster [2] and KaZaA [3] are examples of unstructured networks; and CAN [4], Chord [5], Pastry [6] and Tapestry [7] are four instances of structured networks. In unstructured P2P networks, peers are totally self-organized and resources are randomly placed without any restrictions. Such a network topology is very easy to maintain and is remarkably robust against accidental failures. However, resource location is inefficient due to the lack of global information about the target. Structured P2P networks assign each file a unique key and build a distributed hash table (DHT) that maps each key to a specific peer. The whole network is tightly controlled based on the structured DHT and only supports exact-match searches. Such a network enables efficient searching and guarantees successful discovery of existing data. However, its strict structure makes the network less dynamic and hard to maintain.

This paper focuses on the Gnutella-like unstructured P2P systems. The reasons are twofold: first, Gnutella is one of the few successful and still working commercial P2P systems, and it occupies a fairly large portion of Internet traffic [8,9]. Second, the blind flooding-based search mechanisms produce large volumes of query and reply traffic thus substantially slow down Gnutella, which makes the system far from scalable [10–12].

In a large-scale, open and highly dynamic P2P system, peers could be very heterogeneous in capability (storage, processing power, latency, bandwidth, etc.), availability (the quality of being presented or ready for immediate use [13]) and reliability (malicious peers usually exist, and they can misbehave in many ways [14,15]). Thus, in an attempt to incorporate peers' heterogeneity into the design of a search mechanism, we propose the *Preferential Walk (P-Walk)*, a trust-based probabilistic search mechanism in Gnutella-like unstructured networks. A unique characteristic of our design is the utilization of a trust evaluation method to rate neighbors according to the feedback from previous searches. Neighboring peers assign each other trust ranks. During routing, peers first calculate the forwarding probability of each neighbor according to its trust rank and then preferentially forward queries to the highly ranked neighbors. Note that each peer will continuously update the trust rank it assigns to each of its neighbors through continuously learning from previous searching experiences.

The main contributions of this work are:

- (1) a trust-based probabilistic search mechanism including its core algorithms as well as the trust rank updating procedures;
- (2) the design and implementation of a Gnutella simulator based on the *Barabási–Albert* model;
- (3) an extensive performance evaluation and comparisons of four search algorithms using various metrics;
- (4) the discovery of relationships among peers' trust rank, degree and popularity based on the data analysis from simulation results;
- (5) the P2P view model that provides different views of a real P2P network for an effective search.

2. RELATED WORK

The circumstances when random walk outperforms blind flooding are discussed by Gkantsidis *et al.* [16]. Adamic *et al.* [17] suggest that searching should bias towards the high-degree peers, as they provide server-like functionalities in the networks.



A probabilistic broadcast search protocol combined with a caching mechanism is discussed by Menasce and Kanchanapalli [18]. Simulations show that when the broadcast probability p increases to 0.6, the probability of finding the resource reaches 0.9, and only 10.5% of the peers are involved, whereas when p increases to 1 (namely, blind flooding), the probability of finding the resource increases little, but the percentage of peers involved has increased rapidly to 50%. A similar mechanism called the probabilistic flooding technique (Banaei-Kashani and Shahabi [19]), employs statistical models based on the theory of criticality and complex systems and illustrates that such a search mechanism can improve the efficiency of blind flooding by up to 99%. Both methods highlight the effectiveness of a probabilistic search. Our research objective is different from the reference literature as *P-Walk* applies probability to routing strategy.

3. PROBLEM OVERVIEW

Resources in P2P networks are distributed in the peers. Any peer can issue queries to others to get satisfactory resources (answers). Queries may be in any form such as file identifier or key words with regular expressions [20].

The unstructured P2P overlay can be described as a directed graph where vertices are peers and directed edges are open connections pointing from one peer to another. For a certain query delivered in the network, the peer issuing it is called the *source*, the peer receiving it is called the *receiver* and the peer terminating it is called the *target*. When a peer A receives a query, it first looks up its local storage. If the corresponding resource is found, it will respond to the *source*, and A is the query's *target*; if not, it will decide to which neighbor(s) the query is forwarded to, and such decision-making is called *routing*. The transfer of a query from one peer to another is called a *hop*. The set of peers that the query passes through constitutes a search path, and the number of peers in that set are the number of *hops*. *Hops* have an upper bound called Time To Live (TTL). Once *hops* add one, TTL will be decremented by one, and when TTL decreases to zero, the query will be discarded even if it has not found the satisfactory resources in the network.

The approach used here is to use a simple and practical *routing* strategy to improve search efficiency while simultaneously reducing the large volume of unnecessary traffic.

4. THE DESIGN OF P-WALK

In *P-Walk*, peers assign trust ranks to neighbors. The trust rank for peer B at its neighbor A is a measure of how likely peer A considers that peer B will return the satisfactory answer when forwarding a query to B . The rank value is initialized as zero, and gradually updated based on a one-step feedback mechanism, namely, when peer B replies A with a QueryHit message [2], after validation, A will add one to peer B 's rank value. Algorithm 1 describes the rank initialization and update operation.

Suppose source A has s neighbors N_1, N_2, \dots, N_s ($s \geq 0$), r_i indicates the rank A given to its neighbor N_i . Then, peer A will choose neighbor N_i as its query receiver with the forwarding probability:

$$p_i = \frac{(r_i + r'_i)}{\sum_{n=1}^s (r_n + r'_n)}$$



```

/* Given a peer A, to update the trust ranks of all its neighbors according to the current feedback queue */
Input:  $N_A = \{n_1, n_2, \dots, n_s\}, s, R_A = \{r_1, r_2, \dots, r_s\}, F_A = \{f_1, f_2, \dots, f_m\}$ .
/* Peer A has s neighbors and  $N_A$  is the neighbor set.  $R_A$  is a rank set recording the current values for each
neighbor.  $F_A$  is a feedback queue with the queue size m (m is an integer) recording the currently received
feedback queue from s neighbors of A */
Output: the updated  $R_A = \{r_1, r_2, \dots, r_s\}$ .
for i = 1 to s, do /* retrieving neighbor set  $N_A$  */
  if  $n_i$  is a new coming neighbor
    then  $r_i = 0$  /* initializing the trust rank  $r_i$  of neighbor  $n_i$  in  $R_A$  */
  end if
end for

for feedback  $f_i, i = 1$  to  $m$ , do /* each time, getting the head  $f_i$  from feedback queue  $F_A$  */
  for j = 1 to s, do /* retrieving neighbor set  $N_A$  */
    if feedback  $f_i$  belongs to neighbor  $n_i$  /* matching each neighbor and its feedback */
      then  $r_i \leftarrow r_i + 1$  /* updating the rank of neighbor i on peer A */
    end if
  end for
end for
end for

```

Algorithm 1. Ranking algorithm.

The risk factor r'_i must satisfy the constraint $r'_i \geq -r_i$ ($i = 1, 2, \dots, s$) to ensure that the forwarding probabilities are positive for all neighbors. The non-zero risk factor reflects some instances when a peer's rank cannot truly reflect its behaviors. For example, a malicious peer may be in disguise and behave as a good peer in the beginning; other peers thus regard it as a good neighbor and assign it a relatively high rank. When $r'_i = 0$ ($i = 1, 2, \dots, s$), the forwarding probability for each neighbor is in direct proportion to its rank. Note that $\sum_{s=1}^n (r_s + r'_s)$ converts this rate into a normalized probability, thus, $p_1 + p_2 + \dots + p_s = 1$. Every neighbor has a certain probability to be the receiver. The higher the rank, the higher the probability.

Note that there is no necessary relationship between the rank peer B gives to its neighbor A and the rank peer A gives to B , as the rank value only reflects how one side considers or evaluates the other. For example, A may consider B very dependable, whereas B may think A dishonest.

Algorithm 2 describes *P-Walk*. Figure 1 is a search example where peer A is supposed to issue a query for a certain resource. The query forwarding path follows the direction of the black arrows until finding the corresponding resource at peer J , and the white arrow denotes one-step feedback from J to its predecessor F . Changes of ranks and probabilities for each peer are summarized in tabular part.

5. SIMULATION METHOD

Here we use simulations to evaluate *P-Walk* and compare its performance with three other search mechanisms. Thus, our simulations refer to the following four algorithms.

- (1) *Random walk (R-Walk)*: a peer forwards queries to a randomly chosen neighbor at each hop [16].
- (2) *Max-degree-biased walk (D-Walk)*: a peer forwards queries to the highest-degree neighbor at each hop [17].



```
/* Given a peer A, to calculate the forwarding probabilities for each of its neighbors according to their current
ranks on A */
Input:  $N_A = \{n_1, n_2, \dots, n_s\}$ ,  $s$ ,  $R_A = \{r_1, r_2, \dots, r_s\}$ ,  $R'_A = \{r'_1, r'_2, \dots, r'_s\}$ .
/* Peer A has  $s$  neighbors and  $N_A$  is the neighbor set.  $R_A$  is the current rank set on A and  $R'_A$  records each
neighbor the risk factor */
Output: the forwarding probability set  $P_A = \{p_1, p_2, \dots, p_s\}$ 
sumrank = 0 /* the variable sumrank is initialized and ready for accumulating each neighbors' rank values */
for  $i = 1$  to  $s$ , do /* retrieving neighbor set  $N_A$  */
  for  $j = 1$  to  $s$ , do /* retrieving risk factor set  $R'_A$  */
    if  $r'_j$  is the risk factor of neighbor  $r_i$  /* matching each neighbor and its risk factor */
      then sumrank  $\leftarrow$  sumrank +  $r_i + r'_j$ 
    end if
  end for
end for

for  $i = 1$  to  $s$ , do /* retrieving neighbor set  $N_A$  */
  for  $j = 1$  to  $s$ , do /* retrieving risk factor set  $R'_A$  */
    if  $r'_j$  is the risk factor of neighbor  $r_i$ 
      then  $p_i = (r_i + r'_j) / \text{sumrank}$ 
    end if
  end for
end for
```

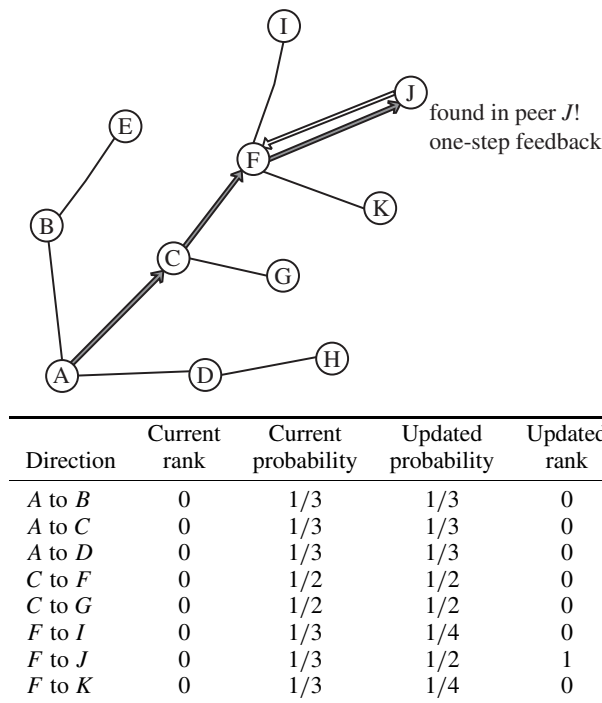
Algorithm 2. Routing algorithm.

- (3) *Max-feedback-biased walk (F-Walk)*: a peer forwards queries to the highest-feedback neighbor at each hop [20].
- (4) *Preferential walk (P-Walk)*: a peer ranks each of its neighbors, and then preferentially forwards queries to the highly ranked neighbor with a high probability at each hop.

5.1. Design of peer structure

In our system, communications between peers form an application-level overlay on top of the physical network. Each peer contains four key function modules:

- (1) *Neighbor Manager*: each peer actively discovers other peers through a *Ping* descriptor. When a peer receives a returned *Pong* message from another peer, it will add the IP address of that peer as its neighbor. In addition, the *Neighbor Manager* is also responsible for endowing each new neighbor with an initial rank and continuously updating the rank once it receives rank submissions from the *Rank Manager*. The risk value queue stores each neighbor's risk value. The risk value for each peer can be randomly generated due to the randomness placement of peers in unstructured P2P networks, or can be calculated based on the searching feedback from each neighbor.

Figure 1. *P-Walk* procedure from source *A* to terminal *J*.

- (2) *Rank Manager*: the *Rank Manager* of a peer receives one-step feedback from the peer's neighbors. It is also responsible for recalculating its neighbors' ranks and submits the new rank values to the *Neighbor Manager*.
- (3) *Resource Manager*: this is, in fact, a local database. The resources are various types of files and each file has a unique, location-independent, global identifier called *fileId*, which can be generated by a hash table or in other ways.
- (4) *Query Manager*: this contains a *Query Generator* and a *Query Database*. The *Query Generator* is responsible for producing a random query sequence continuously. Each query has several fields such as *fileId* (for simplicity, we use each file's *fileId* to label its corresponding query), *ttl*, *path* (a record of the query's routing path), *succTag* (a symbol indicating whether this query has successfully found its corresponding file). The *Query Generator* submits these queries to the *Query Database* while at the same time forwarding them one by one to the peer's neighbors. The *Query Database* updates the queries' settings once receiving these queries' responses from other peers.

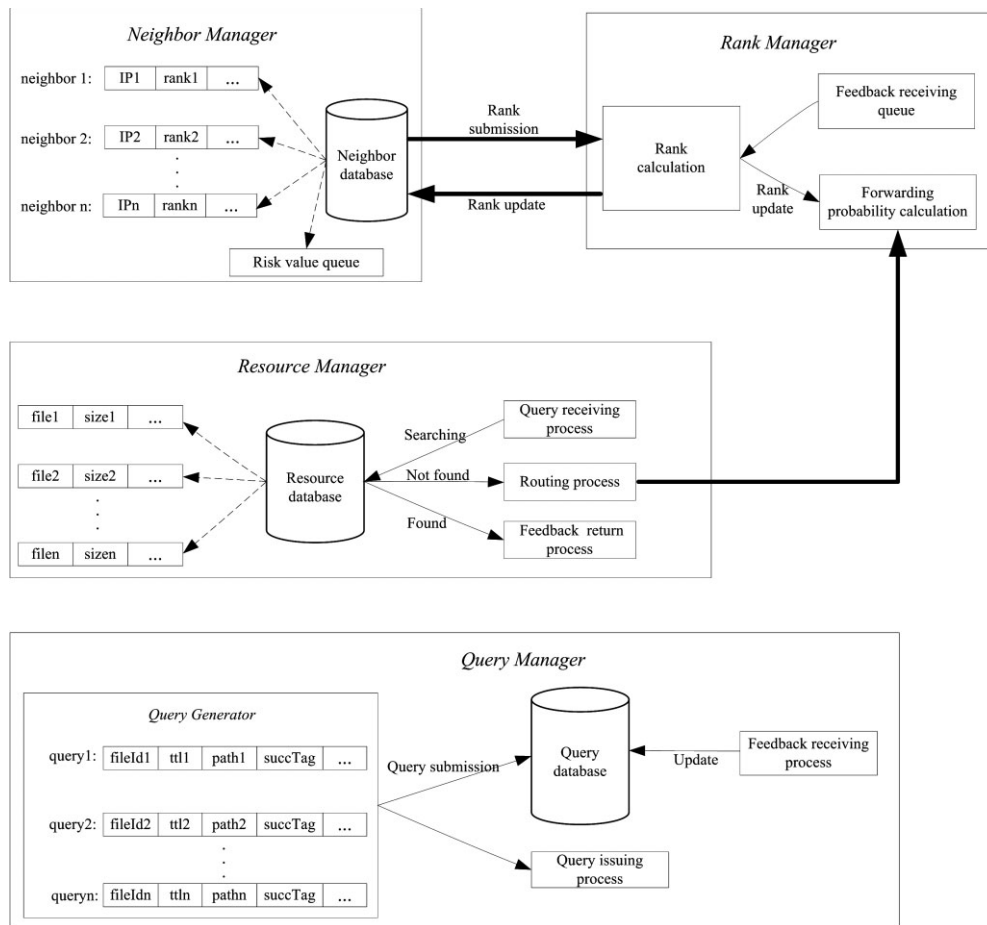


Figure 2. Four function modules of each peer.

We present the schematic illustration of each function module as shown in Figure 2. The dashed arrow line represents ‘part of’, for example, the risk factor queue is stored in *Resource database*, while the solid arrow line represents communications between two components within one peer.

5.2. Simulation setup

We deploy a fixed Gnutella topology where peers are static without joining and departing. This seems ideal compared with real systems. However, suppose that all the searching processes are complete



before the change of the topology of the network, our performance results are still valuable in a practical sense. In addition, the crawler trace in [13] shows that peers' departing and joining rates are roughly equal thus the total number of peers in the network remains constant.

Our simulations are based on the observed data of Sripanidkulchai [12], Ripeanu and Foster [21] and Schlosser and Kamvar [22]. We will consider three aspects: network degree distribution, file popularity and query popularity. The degree distribution is a statistical property for a network topology denoting the number of peers that have i ($i = 0, 1, 2, \dots, n$) open connections. By using the *Barabási–Albert* model with parameters $q = 0.125$, $\lambda = 0.01$ and 1000 fixed peers [23,24], we generate a two-segment power law degree distribution similar to the trend observed by Ripeanu and Foster [21]. In addition, peers in our network may be good or malicious where malicious behavior is based on Mishra [14]. As the sum of each peer's risk value r' and its rank r should be a positive integer the risk factor for each peer is generated randomly with a value between 0 and 0.5.

File popularity denotes the number of peers that have the same file in their *Resource Managers*, while query popularity denotes the number of peers that search for the same file [25]. Thus, if one file is stored in n peers, it has n file replicas in the network. Accordingly, if one query issued n times from n peers, it has n query replicas in the network.

For a particular file, suppose its file popularity is in directly proportion to its query popularity. For 10000 files, we use the *Barabási–Albert* model with $q = 0.1$, $\lambda = 4$ to generate file replicas which follow a log-quadratic distribution [22]. The total number of files together with their replicas reaches 39458 files and the most popular file has 155 replicas. We randomly distribute these file replicas among 1000 peers according to the peers' degree. High-degree peers have higher capability to store more replicas. Each peer randomly generates a total of 200 queries and issues one query at a time. Therefore, the whole system issues 1000 queries at a time. Note that each query cannot find its corresponding file in its *source* peer.

5.3. Metrics

We evaluate the efficiency of the four search mechanisms according to the following metrics.

- (1) *Success rate*: the ratio of successfully completed queries (finding the answer within a predefined TTL) to the total number of queries.
- (2) *Response time*: the number of hops a successfully completed query passes through.
- (3) *Effectiveness against malicious behavior*: malicious peer percentage is defined as the ratio of the number of malicious peers to the total number of peers in the system. We first show how the success rate and response time change as the malicious peer percentage increases. Then, we record the average number of messages each peer needed to operate each time to examine whether the *P-Walk* can isolate malicious peers.
- (4) *Self-learning ability*: this is measured by collecting 100 queries issued each time, and averaging the number of hops they pass through in finding resources.

6. PERFORMANCE EVALUATION

6.1. Success rate and response time

Figure 3 shows the success rate of four search algorithms as TTL varies from 5 to 100 with a malicious peer percentage of 10%. The *P-Walk* achieves the highest success rate, followed sequentially by

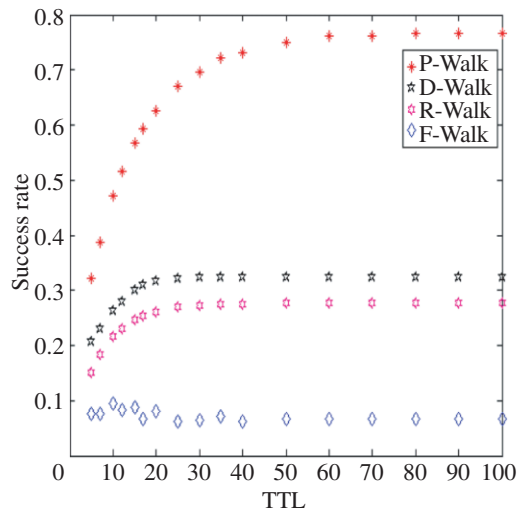


Figure 3. Success rate as TTL varies.

D-Walk, *R-Walk* and *F-Walk*. Adamic *et al.* [17] explained in detail why *D-Walk* is better than *R-Walk*. *F-Walk* is a kind of deterministic search in that routing is always towards the maximal feedback peers. As malicious peers misbehave themselves and accumulate feedback rapidly in the beginning, *F-Walk* is inevitably misled by malicious behavior.

We then evaluate the success rate from another perspective. Figure 4 shows the trends of the success rates as malicious percentage increases in the network. The four trend curves correspond to the four algorithms when TTL is seven, and the relative trends under other TTL are similar. For the same reason explained above, the *P-Walk* outperforms the other three.

After all 200 000 query-routing processes ended, we calculate how many queries have successfully completed at each TTL-hop. As illustrated in Figure 5, the number of successful queries our *P-Walk* found within any TTL-hop is the highest among the four algorithms except that *D-Walk* exhibits a little more than *P-Walk* in the first TTL-hop. At the beginning of the simulation, *P-Walk* is equivalent to *R-Walk* because all of the peers' ranks are still at their initial values. On the other hand, *D-Walk* has a bias towards high-degree peers that have more resources, thus it can find more resources in the first TTL-hops. However, in the following TTL-hop, *D-Walk* is obviously not as good as *P-Walk* as it does not consider feedback information from previous searches.

6.2. Effectiveness against malicious behavior

One important problem is the system's robustness against malicious behavior. A well-designed mechanism can effectively isolate malicious peers. When forwarding messages between peers, one peer can intentionally reduce the messages forwarded to those it thinks malicious. So, an effective

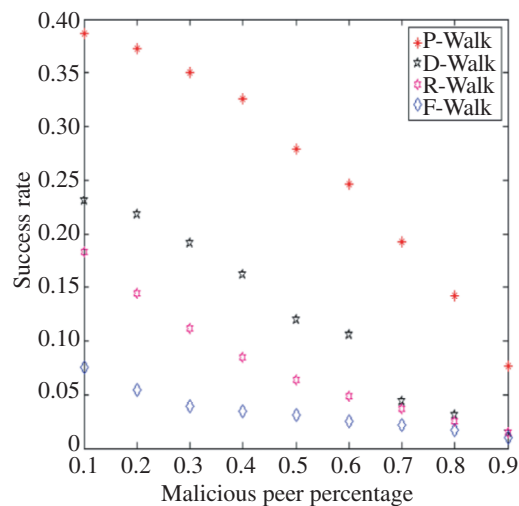


Figure 4. Success rate as malicious peer percentage varies.

mechanism can achieve the goal that the average number of messages that the good peers operate which is far greater than that of a malicious peer. In addition, as the simulation runs, the number gap between the good and malicious peers increases, that is to say, this system can gradually isolate malicious peers and effectively prevent malicious behavior. Based on the above consideration, we record the number of messages forwarded to each peer at a time.

In Figure 6, the y -coordinate denotes the number of messages each peer needs to operate and the x -coordinate denotes time. We perform four groups of simulations under different TTL for comparison. When routing, *P-Walk* can effectively decrease the number of messages forwarded to the lowly ranked neighbors, consequently minimizing malicious behavior to a large extent as the simulation runs. *R-Walk* treats peers equally without discrimination, so the numbers of messages on malicious peers are generally equivalent to that on good peers. However, the other two algorithms exhibit a worse ability to distinguish between good peers and malicious peers, as malicious peers often disguise themselves as good peers in the beginning in order to gain enough visits and feedback. Therefore, the message numbers on malicious peers quickly increases and becomes clearly larger than that of good peers.

6.3. Self-learning ability

With the increasing number of messages delivered from one peer to another, neighbors continuously update the ranks they assign to each other and gradually accumulate experience such as which neighbor(s) has (have) offered more positive replies. Thus, search procedures will be increasingly efficient using the rank as a guide to direct routing. We call such an ability self-learning.

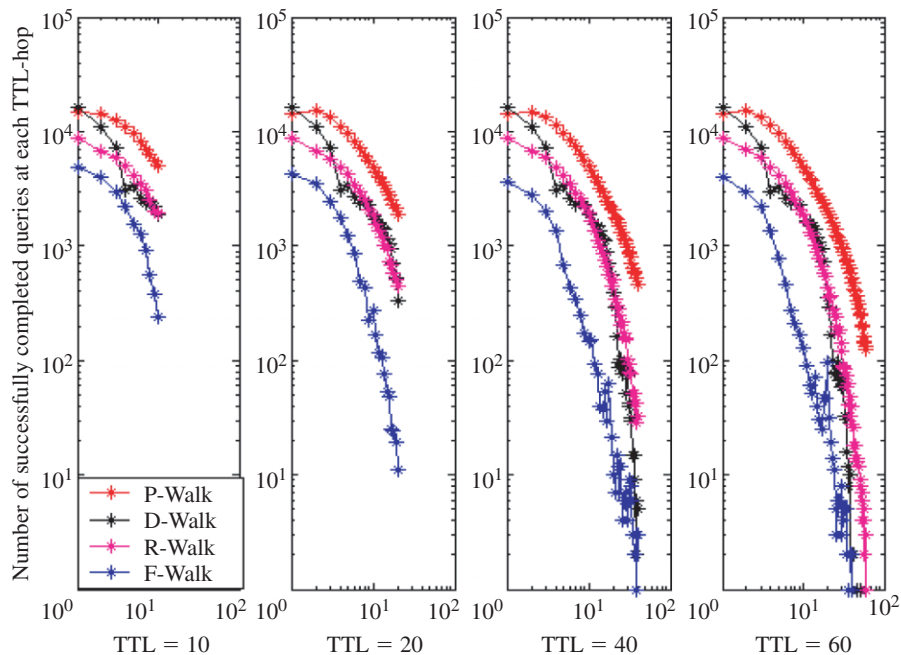


Figure 5. Number of successfully completed queries at each TTL-hop as TTL varies.

The four algorithms described above all adopt previous experience as a guideline (note that *R-Walk* in Gnutella-like networks exhibits natural deviations towards the high-degree peers, see Adamic *et al.* [17]). In order to evaluate how the self-learning ability facilitates the search procedure, we gather the query messages issued at the same time and calculate their average hops under various TTL. As Figure 7 demonstrates, although the queries' average hops in the four algorithms all gradually drop as time passes, they exhibit different self-learning abilities. *P-Walk* is the fastest self-learner, *R-Walk* and *D-Walk* take second and third places respectively, and *F-Walk* is the slowest self-learner. Clearly, *P-Walk* is able to make relatively more accurate estimations about which neighbor queries should be forwarded to while the other three algorithms are easily misguided by malicious behavior.

When *P-Walk* is working, the latterly issued queries use clearly shorter hops to find resources than those issued formerly, which reduces the large volume of query and reply traffic. Consequently, users will get quicker response when the system becomes mature.

6.4. Ranking analysis

Recall that for one peer, the ranks given from different neighbors are independent, and the ranks that one peer assigns to its neighbors have nothing to do with the ranks those neighbors assign to it.

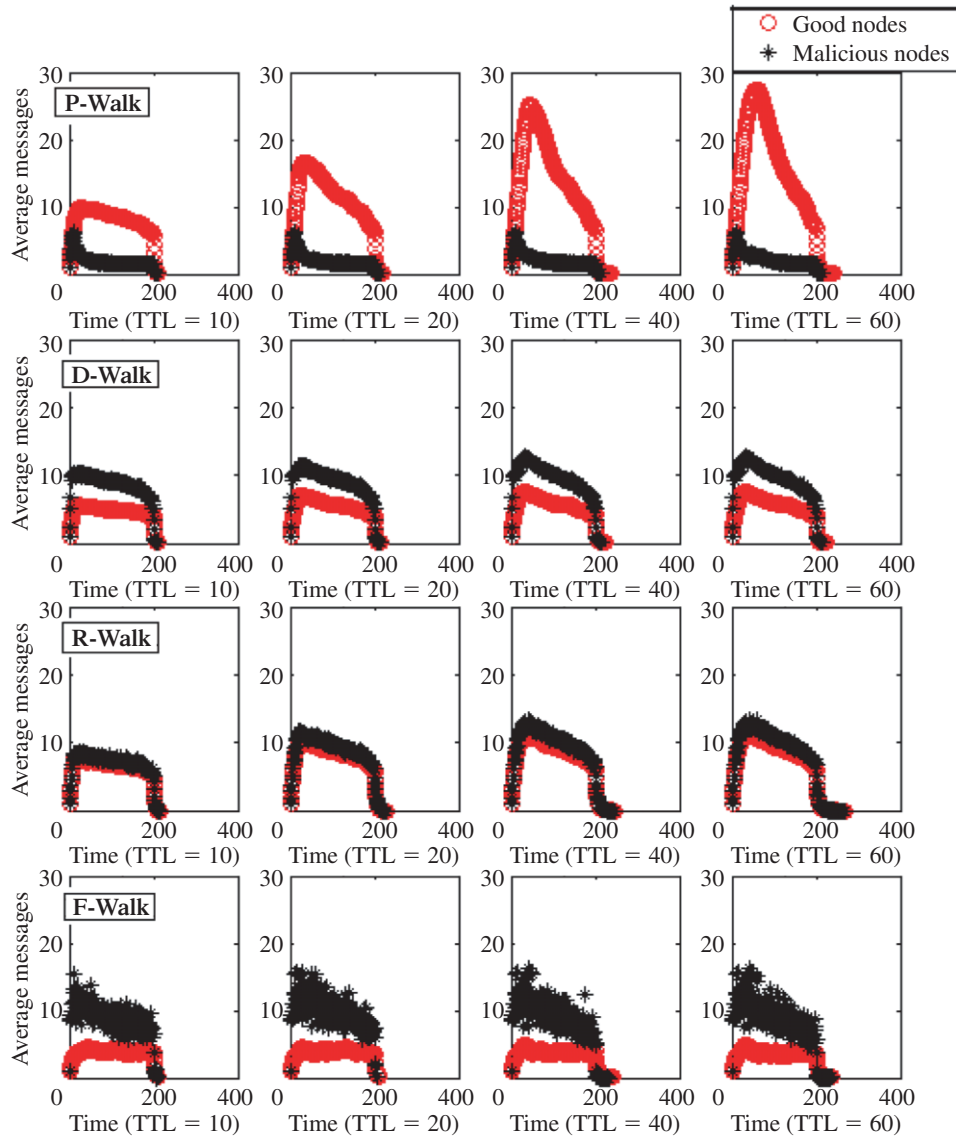


Figure 6. Average numbers of messages peers (good and malicious) operate each time.

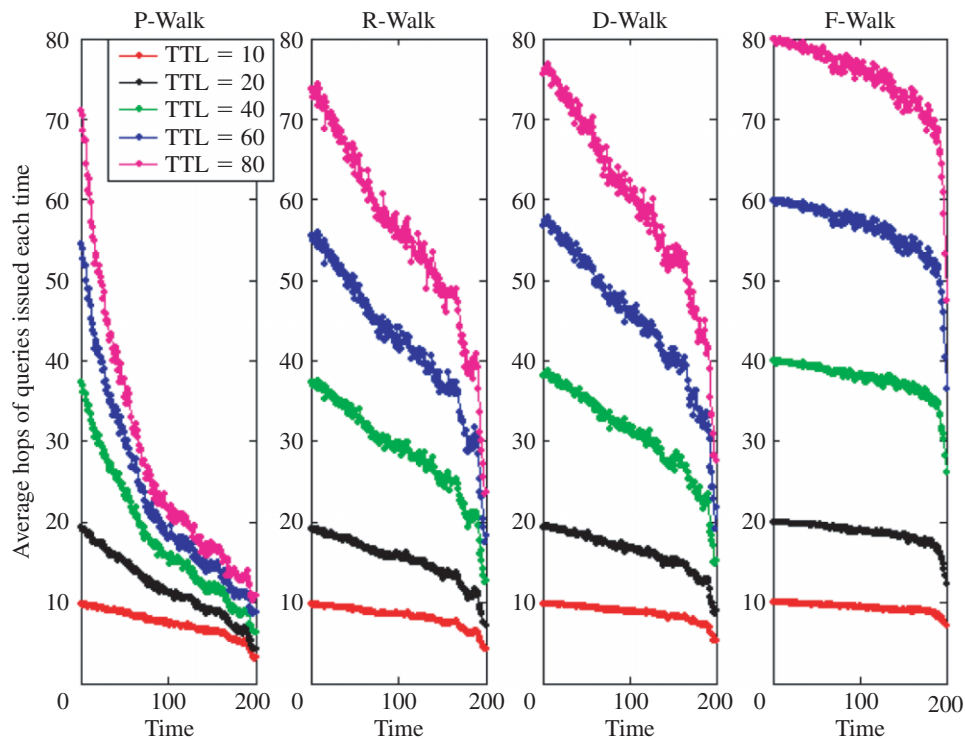


Figure 7. Self-learning ability of the four algorithms.

In order to get an insight into the relations between peers and their ranks, we collect the rank values in each peer's *Neighbor Manager* after all 200 000 queries have been completed. A rank distribution is defined as the number of peers whose ranks fall in the same intervals. Suppose that one peer P and n ranks from its n neighbors r_1, r_2, \dots, r_n , three measures can be used to evaluate each peer's rank value: summing these ranks $r_{\text{sum}} = \sum_{i=1}^n r_i$; calculating the average $r_{\text{avg}} = \sum_{i=1}^n r_i / n$; and only focusing on the maximal value $r_{\text{max}} = \max\{r_1, r_2, \dots, r_n\}$. Thus, the rank distribution can denote the summing rank distribution, the average rank distribution or the maximal rank distribution. The three insets of Figure 8 illustrate an approximate linear relationship between each peer's rank and its degree, and the three lines have a similar trend. In general, the higher degree one peer has, the higher the rank, and the high-degree peers are in the minority. Figure 9 shows that the three kinds of rank distributions all exhibit power-law-like trends when plotted on a double-logarithmic scale. These observations accord with what we expect. In real systems, there are only a small number of high-degree peers with server-like characteristics but a large number of peers are client-like. High-degree peers have a higher capability [15]. As demonstrated in our simulations, the combined performance of one peer (including capability, availability and reliability) can be reflected by the rank values given

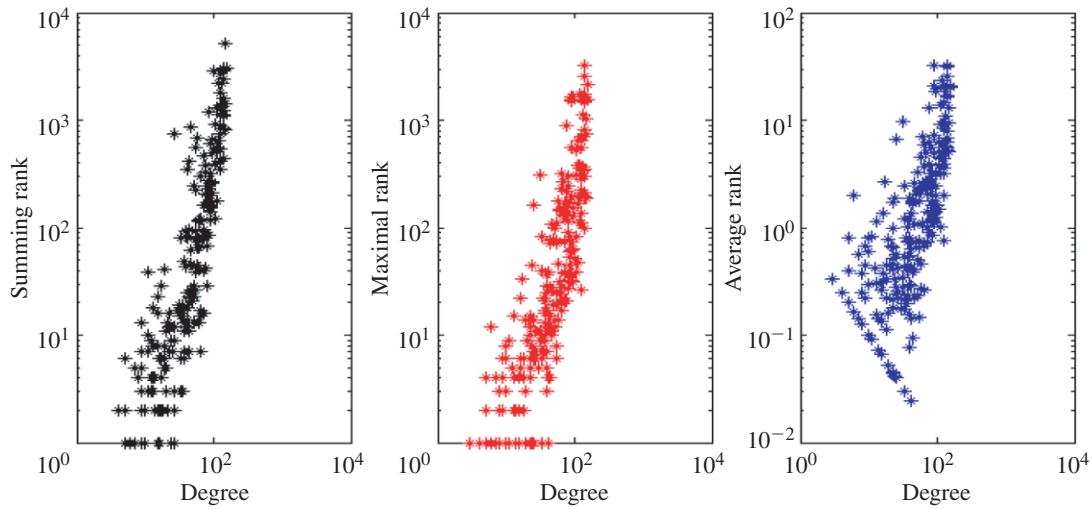


Figure 8. The relationships between each peer's rank (i.e. summing rank, average rank and maximal rank) and its degree.

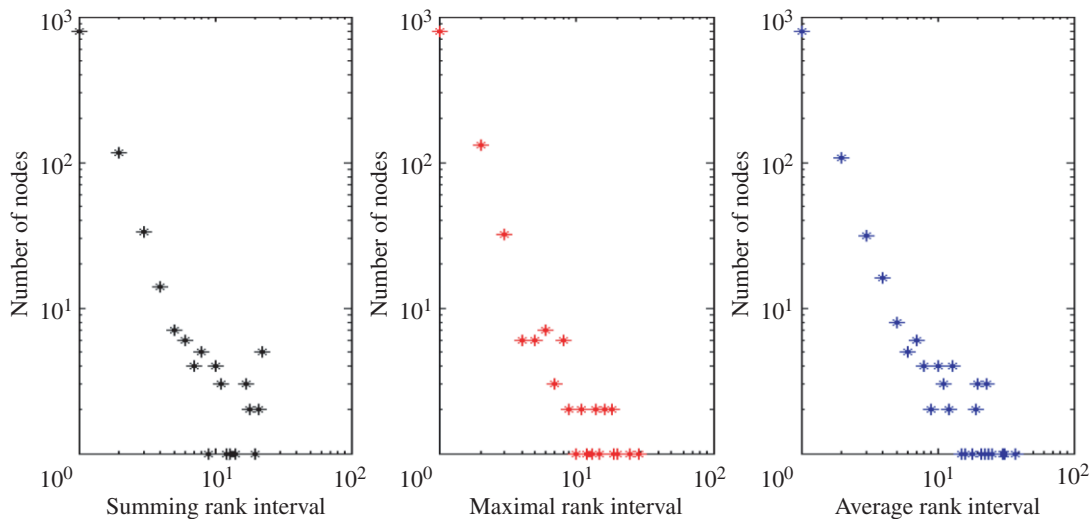


Figure 9. The summing rank distribution, the maximal rank distribution and the average rank distribution.

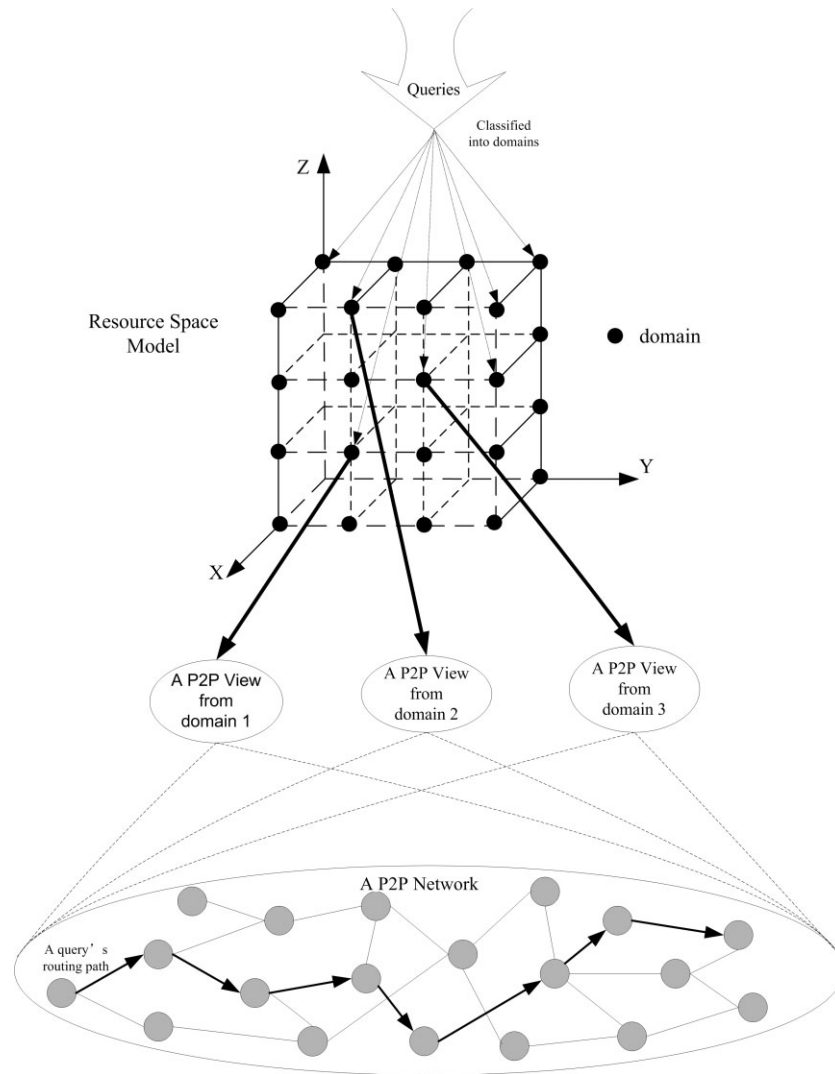


Figure 10. Search in different views of the same P2P network.



from the peer's neighbors. If each peer's rank follows a positive linear relationship with its degree, the rank distribution exhibits the same trend with respect to the degree distribution.

7. P2P VIEWS

A P2P network usually contains information of various types, so it could answer questions of different domains (different interests or applications). The network will form different views where a peer could play different roles with different ranks. Figure 10 shows the principle of the P2P views. An arriving query will be located in an appropriate view, which is responsible for query routing and interaction between the view and the real P2P network. The proposed *P-Walk* is effective in each view if multiple views are involved.

The Resource Space Model [26] provides an effective means for classifying and managing the views of large amounts of versatile information. Each point in the space corresponds to a view. The theories of normal forms and integrity constraints ensure the correctness of view management.

This P2P view model is also in line with our daily life social network, which contains a large amount of information and knowledge on society, science and technology. A query is routed only in a view of the entire social network. The P2P view, together with the *P-Walk* approach, can be a model of social networks.

8. CONCLUSIONS

The trust-based probabilistic search mechanism *P-Walk* is based on a loosely controlled Gnutella-like P2P network. Neighboring peers assign a trust rank to each other through the feedback from mutual message delivery. Peers preferentially forward query messages to the highly ranked neighbors who are considered most likely to return the satisfactory resources. Extensive simulation results show that *P-Walk* can significantly improve the query success rate and response time while at the same time effectively reducing traffic overhead and controlling malicious behavior. The simplicity and robustness of *P-Walk* makes it feasible to be directly deployed in real systems. Considering the diversity of resources shared in the P2P networks, we build a P2P view model to provide domain views where a peer can play different roles with different ranks. Such a model further facilitates resource sharing and retrieval in P2P networks.

The proposed mechanism is an important part of the P2P overlay in the Knowledge Grid environment [26,27]. Work is ongoing to apply the proposed approach to realize a semantic P2P overlay in the environment to support intelligent applications.

ACKNOWLEDGEMENTS

The authors thank all of the team members of the China Knowledge Grid Research Group (<http://kg.ict.ac.cn>) for their help and cooperation.

REFERENCES

1. KaZaA. <http://www.kazaa.com> [30 April 2005].
2. The Gnutella protocol specification 0.6. <http://rfc-gnutella.sourceforge.net> [25 April 2005].



3. Napster. <http://www.napster.com> [30 April 2005].
4. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. *Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, San Diego, CA, 2001. *ACM SIGCOMM Computer Communication Review* 2001; **31**(4):161–172.
5. Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Lecture Notes in Computer Science*, vol. 2218). Springer: Berlin, 2001; 329–350.
6. Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking* 2003; **11**(1):17–32.
7. Zhao B, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz J. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* 2004; **22**(1):41–53.
8. Saroiu S, Gummadi PK, Dunn RJ, Gribble SD, Levy HM. An analysis of Internet content delivery systems. *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, December 2002. USENIX Association: Berkeley, CA, 2002; 315–327.
9. Sen S, Wang J. Analyzing peer-to-peer traffic across large networks. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, Marseille, France, November 2002. ACM Press: New York, 2002; 137–150.
10. Liu YH, Liu XM, Xiao L, Ni LM, Zhang XD. Location-aware topology matching in P2P systems. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 4, Hong Kong, China, March 2004. IEEE Press: Piscataway, NJ, 2004; 2220–2230.
11. Ritter J. Why Gnutella can't scale. No, really. <http://www.tch.org/gnutella.htm> [25 April 2005].
12. Sripanidkulchai K. The popularity of Gnutella queries and its implications on scalability. <http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html> [30 April 2005].
13. Bhagwan R, Savage S, Voelker GM. Understanding availability. *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, 2003 (*Lecture Notes in Computer Science*, vol. 2735). Springer: Berlin, 2003; 256–267.
14. Mishra M. Cascade: An attack resistant peer-to-peer system. *Proceedings of the 3rd New York Metro Area Networking Workshop (NUMAN)*, New York, September 2003. Available at: <http://www.nyman-workshop.org/2003/2003.html>.
15. Saroiu S, Gummadi PK, Gribble S. A measurement study of peer-to-peer file sharing systems. *Proceedings of Multimedia Computing and Networking (MMCN)*, vol. 4673. SPIE: Bellingham, WA, 2002; 156–171.
16. Gkantsidis C, Mihail M, Saberi A. Random walks in peer-to-peer networks. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, Hong Kong, China, March 2004. IEEE Press: Piscataway, NJ, 2004; 130–140.
17. Adamic LA, Lukose RM, Puniyani AR, Huberman BA. Search in power-law networks. *Physical Review E* 2001; **64**:46 135–46 143.
18. Menasce DA, Kanchanapalli L. Probabilistic scalable P2P resource location services. *ACM SIGMETRICS Performance Evaluation Review* 2002; **30**:48–58.
19. Banaei-Kashani F, Shahabi C. Criticality-based analysis and design of unstructured peer-to-peer networks as 'complex systems'. *Proceedings of the 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, Tokyo, Japan, May 2003. IEEE Press: Piscataway, NJ, 2003; 351–358.
20. Yang B, Garcia-Molina H. Improving search in peer-to-peer networks. *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002. IEEE Computer Society Press: Los Alamitos, CA, 2002; 5–14.
21. Ripeanu M, Foster I. Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems. *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (Lecture Notes in Computer Science*, vol. 2429). Springer: Berlin, 2002; 85–93.
22. Schlosser MT, Kamvar SD. Availability and locality measurements of peer-to-peer file systems. *Scalability and Traffic Control in IP Networks (Proceedings of ITCOM)*, vol. 4868, Boston, MA, July 2002. SPIE: Bellingham, WA, 2002; 310–321.
23. Barabási AL, Albert R. Emergence of scaling in random networks. *Science* 1999; **286**:509–512.
24. Krapivsky PL, Redner S. A statistical physics perspective on Web growth. *Computer Networks* 2002; **39**:261–276.
25. Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. *Proceedings of the 16th ACM International Conference on Supercomputing (ICS)*, June 2002. ACM Press: New York, 2002; 84–95.
26. Zhuge H. *The Knowledge Grid*. World Scientific: Singapore, 2004; 99–138.
27. Zhuge H, Sun XP, Liu J, Yao EL, Chen X. A scalable P2P platform for the knowledge Grid. *IEEE Transactions on Knowledge and Data Engineering* 2005; **17**(12):1721–1736.