A Semantic Overlay Network for Unstructured Peer-to-Peer Protocols

Junfeng Xie, Zhenhua Li and Guihai Chen
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, P. R. China
{jfx, lizhenhua}@dislab.nju.edu.cn, gchen@nju.edu.cn

Abstract

Peer-to-Peer computing has become a popular networking paradigm for file sharing, distributed computing, collaborative working, etc. The widely used unstructured Peer-to-Peer protocols mainly face two problems affecting their working efficiency: 1) inefficient flooding-based search, 2) topology mismatch between the overlay network and its underlying network. In this paper, we propose to organize nodes into a semantic overlay network called CON which is composed of special interest groups based on nodes' contents. CON guides the content search with semantic information so that it avoids most of the flooding cost. In order to alleviate the mismatch problem, nodes in CON initialize links according to their underlay proximity. Simulation results show that our mechanism efficiently increases the query success rate and reduces the traffic cost and query latency. We also compare CON with the similar work, which illuminates that CON performs better in many aspects.

1. Introduction

In recent years, Peer-to-Peer systems have been widely used as the base infrastructures of many Internet applications, such as resource sharing (*e.g.*, BitTorrent [1], eDonkey [2]), collaborative working (*e.g.*, Groove [4]), distributed computing (*e.g.*, SETI@home [7]) and so on. P2P systems are mainly organized into three kinds of architectures: centralized, decentralized unstructured and decentralized structured [16]. Centralized P2P systems, such as Napster [6], use index servers to maintain a directory for shared files of peers so that a participant can search for the location of the desired content from those servers. The query initiator then downloads files directly from the *peers* (rather than *servers*) that hold them. Unfortunately, these servers turn out to be system bottlenecks and also make the system vulnerable to malicious attacks. On the contrary, the decentralized protocols do not rely on central servers for data retrieval and offer much better scalability and resilience. Structured systems, such as Chord [25] and CAN [20], use distributed hash table (DHT) to strictly organize the placement of data and the network topology. Unstructured systems, such as Gnutella [3] and KaZaA [5], place documents and nodes randomly, without correlation with the network topology. Unstructured systems exhibit ease in topology maintenance but face two main problems that seriously degrade their performance [22] [23]:

1) *topology mismatch* between the overlay network and its underlying network,

2) *inefficient flooding-based search* for desired content.

This paper focuses on the possible performance improvements of decentralized unstructured P2P systems. We deal with the first problem by initializing overlay links according to the underlay proximity when nodes bootstrap and the flooding-based search is based on it. For the second problem, we intend to forward the query to the most relevant nodes by introducing a table called *file table* appended to each file. Its entries point to some of the nodes that also have this file, thus forming a SIG (special interest group) corresponding to this file. In this way, a semantic overlay network called CON (*content overlay network*) is embedded into the existing overlay network. Now two kinds of complementary search mechanisms are available: one is based on SIG, the other on flooding. The CON-directed search has *high performance* for it only queries peers with similar interests and the flooding-based search is executed at *low cost* in that the neighbors are connected with low latency. In our system the flooding-based search is just used as a complement to the CON-directed search, *i.e.*, it is triggered as the compensation only when the scale of the CON-directed search is not big enough. Besides, a node can belong to multiple SIGs, and a SIG is also

978-1-4244-1890-9/07/$25.00 ©2007 IEEE

able to adjust itself as nodes frequently join and leave the system.

Simulation results show the improvement brought by our method. In a Waxman-model network [27] initially with 2,500 nodes and 25,000 files, as file tables are gradually filled, the latency per query decreases by 78% and the query success rate increases by 50%. Furthermore, the traffic cost is also reduced by 56% and the topology match is indeed improved for the stretch decreases by 77%. Besides, with the same query success rate, the average latency of the CON-directed search is only 40% of the previous similar *shortcut-based search* proposed by Sripanidkulchai et al. [24] and the stretch is 50% of that.

The rest of this paper is structured as follows. Section II reviews the related work. Section III presents the detailed system design. Section IV is the performance evaluation and Section V discusses more about CON. Finally, we conclude the paper in section VI.

2. Related work

2.1. The search mechanisms

To the best of our knowledge, the work most related to ours is due to Sripanidkulchai et al. [24]. They use a *shortcut list* to record the recently visited nodes from which files are downloaded and rank the shortcuts based on some criterions. The rationale is also to cluster nodes based on their interests for they realize the existence of the so-called *interest-based locality*. The query initiator first checks whether the nodes in its shortcut list have the desired file and if not, turns to the flooding-based search. In this sense, such one-hop strategy is essentially different from our multi-hops CON-based strategy. The other fundamental differences between the two systems are the perception of *similar interest* as well as the mechanism to cluster nodes.

Another perspective introduced by Tang et al. [26] is not to cluster nodes with similar interest but to cluster documents with similar semantics in identifier space. It is intended for the structured protocol CAN [20] in that they observe the natural correspondence between the topology of CAN and the document's latent semantic indexing (LSI). LSI is used as a document's identifier for the insertion and retrieval of that document. As a result, the relevant documents are placed close to each other in the identifier space, therefore forming a sort of document semantics cluster.

Some other work goes to great length to make the selection of next hop more effectively. Random walk proposed by Lv et al. [16] has better performance than the flooding-based search in a Zipf-like or small-world graph. Gia [8] uses biased random walk to forward incoming query to its neighbor who has the highest capacity. Such choice is based on a simple intuition: the more documents a node has, the more likely it will satisfy query requirements. The overloads caused by such biased selection is tackled by the use of flow control token.

On contrary to Gia [8], Crespo et al. [9] try to incorporate neighbors' semantic information to guide the query forwarding direction. *Routing index* is used to store the number of documents on each topic along the path via each neighbor and the query will be forwarded to the neighbor with the largest number of relevant documents. Such greedy mechanism guarantees to send the query to the most promising candidate among its neighbors, but the maintenance cost is non-negligible.

Like Crespo et al. [9], to make search more intelligent, Kalogeraki et al. [12] add semantics into the search mechanism. The most recent queries passed by each neighbor is recorded in a node's corresponding *peer profile*. When a new query comes to that node, by calculating the similarity between the new one and the recorded ones, the neighbor satisfying the query with the highest probability can be determined.

2.2. Topology mismatch

Researchers have proposed a number of methods [21, 19] to alleviate the topology mismatch problem of overlay networks. As classified by Gummadi et al. [10], the proposed methods generally fall into three categories: Proximity Neighbor Selection (PNS), Proximity Route Selection (PRS) and Proximity Identifier Selection (PIS). In our CON overlay network, nodes initialize links according to underlay proximity when they bootstrap. Therefore, our method to alleviate topology mismatch belongs to Proximity Neighbor Selection (PNS).

The *supernode* method is popularly utilized in many practical P2P systems, such as KaZaA [5] and the Gnutella protocol version 0.6 [3]. Experiments by Liang et al. [13] show that the mismatch problem is indeed alleviated by using supernodes.

Liu et al. [14] propose *LTM* scheme in which each peer issues a detector message so that the peers receiving the detector can record relative delay information. Based on it a receiver detects and cut unnecessary links. The major drawback is that all peers need to be synchronized so LTM requires the support of the Network Time Protocol. Alternatively, Xiao et al. [28] propose *ACE* in which every peer builds an overlay MST among itself and its neighbors and then

optimizes the neighbor connections that are not on the tree. However, ACE can only work with one-hop logical neighbors, and the convergent speed is relatively slow. As an improvement, Liu. et al. [15] introduce *SBO* to optimize the overlay topology by identifying and replacing the mismatched connections. It has fast convergent speed and does not need Network Time Protocol, therefore overall, SBO outperforms ACE and LTM.

All the technologies mentioned above *adjust* the overlay connectivity after its construction. Our method, by comparison, takes topology-aware issue into consideration while *constructing* the overlay network.

3. System design

For each file, We intend to construct a SIG composed of all the nodes that have *this file* therefore embed the *file table* in the file, shown in Table 1. Each IP address (in fact, it also includes the node's id, port number, etc) represents a SIG member, *i.e.*, the node which also has *this file*. n is not a fixed number but it is always small (usually $n \leq 5$) so that these n nodes are only a small proportion of the whole SIG. Because of the high dynamic of P2P network, the system prepares backup nodes for each entry.

Table 1. file structure

file	**file table**			
content	IP_1	IP_2	...	IP_n

To enable the CON-directed search, we utilize VSM to represent files and queries as vectors in a Cartesian space. In information retrieval community, various IR techniques are proposed to represent contents. We adopt the most popular and well-studied statistical IR algorithm, vector space model (VSM) [17]. For the file, the vector is used as its identifier and for the query, it is the search criteria. The *similarity* between a query and a file is measured as the cosine of the angle between their vector representations:

$$similarity(q, f) = \cos < \vec{q}, \vec{f} >= \frac{\vec{q} \cdot \vec{f}}{|\vec{q}||\vec{f}|}$$

where $\vec{q}$ and $\vec{f}$ represent the vectors of the query q and file f, respectively.

3.1. The solution to topology mismatch

Nodes initialize overlay links according to underlay proximity when they bootstrap. The overall target is to initialize the overlay layout consistent with its underlying network. A newly joining node first contacts a well-known server to get a list of nodes near itself. However, the well-known server only knows a limited number of nodes. As an improvement, the newly joining node then contacts its neighbors' neighbors to adjust its neighbor set in that these neighbors should be near as well. This procedure can be performed recursively several times, but the bandwidth consumption will also be increased greatly. According to our simulation experience, contacting neighbors one or two hops away is good enough. Besides, a node periodically detects its neighbors and its neighbors' neighbors to adjust its neighbor set.

3.2. The design and utilization of CON

We need three thresholds in the search mechanism.

1. **Maximum-candidate threshold** (T_c) controls the number of candidate peers to forward the query.
2. **Minimum-similarity threshold** (T_s) defines the low bound of the similarity between the query and candidate peers to forward the query.
3. **Relevance threshold** (T_r) determines whether a file is relevant enough to be returned to the initiator. It is introduced to return not only exactly the same file requested by the query but also the very relevant ones.

Suppose a node wants to search for the A.M. Turing biography. It first creates a VSM vector for this query, then calculates the similarity between this query and each file it stores. The files with similarity beyond T_s are picked and they are sorted by descendant similarity. We choose the first T_c ones and because these files (*e.g.*, the John von Neumann biography) are *the most relevant ones* to this query, their corresponding SIGs are regarded as the most promising candidates that should have the desired file. Therefore the query is sent to the members of these SIGs. If there are not enough (*i.e.*, less than T_c) relevant files which may happen especially when the initiator is a new comer and has few files, the flooding-based search in overlay network is triggered to compensate.

The intermediate nodes that receive the query decrease the TTL by 1 and forward it in the same way. At the same time, their files whose similarity with the query are beyond T_r are sent to the query initiator. However, when the query's TTL is 0, it is no longer forwarded.

The query initiator regards the search is finished after a certain period, then it ranks all the received files and returns them to the user. Besides, it joins the SIG by notifying the nodes of the file table its existence and these nodes deploy the FIFO strategy to insert the initiator into their relevant file tables.

To rank the table entries, let us first introduce two concepts:

Definition 1 (Similarity(f_1, f_2))

$$similarity(f_1, f_2) = \cos < \vec{f_1}, \vec{f_2} >= \frac{\vec{f_1} \cdot \vec{f_2}}{|\vec{f_1}||\vec{f_2}|}$$

where f_1 and f_2 are two files whose vectors are $\vec{f_1}$ and $\vec{f_2}$.

Definition 2 (Similarity(n, f)) *n is a node and f is a file and $Similarity(n, f)$ is the number of files with $similarity(f_i, f)$ beyond T_r in node n. Intuitively, it represents the relevance degree between n and f.*

For file f, the order of table entries is determined by their corresponding *similarity(n, f)* where n is the node pointed. In this way, the entries are ranked based on their relevance to its file and our search mechanism is modified to only choose the first entries in the table as candidates. This enhancement requires some bandwidth consumption to calculate the *similarity(n, f)* as well as an additional field in each table entry to record it, but the advantage is twofold: First of all, such order enables a query to converge quickly in the SIG graph to nodes with the largest number of relevant files, thus raising the success rate with fewer hops. The other benefit is about motivation: the more files on a particular topic a user downloads, the higher her node ranks in other SIG tables, therefore the more likely she will serve others.

Lv et al. [16] have proven that the *square-root replication* optimizes the average search size as well as the utilization rate. This strategy replicates a file proportional to the square root of its query probability. To archive this, $c\frac{n}{r_i}$ replica (where c is a constant, n is the total number of nodes in the system and r_i is the number of a file's replica) should be created after the query initiator downloads its desired file. Lv et al. [16] have proven that in the random walk system, it is just the same with the number of probes it took before finding the file. In our simulation, however, we will use this formula directly and c is set as 0.01, r_i is the order of the file's SIG graph, gotten by a DFS search and n is set as 2,500. The distribution of these new replica is as follows. Suppose the desired file is f and r replica are created. The initiator ranks the nodes along the search path according to their *similarity(n, f)* and the first r ones are regarded as the candidates to store f. Such strategy causes the *Matthew effect* based on the rationale that the more files on a particular topic (*e.g.*, the biographies of computer scientists) a user stores, the more likely she will be interested in that replicated one(*i.e.*, the A. M. Turing biography) which she does not already have. On the other hand, in the light of the entry order, the candidates are also those most likely to serve other relevant queries, so these replica also increase their success probability to satisfy them.

A subtle problem is that several SIGs of *the same file* may develop independently without any knowledge of each other's existence, in that the file insertion is not broadcasted or some nodes may crash unexpectedly.

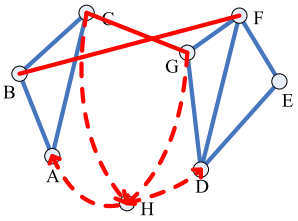

Figure 1. two independent SIGs of the same file

To make this phenomenon more evident, Fig. 1 shows an example. Note that only CON links are showed. Two independent SIGs (*i.e.*, the one consisted of node A, B and C and that consisted of node D, E, F and G) of the A.M. Turing biography exist. They do not know each other so if a query (initiated by node H) for the Turing-relevant content is sent to node A, it is only forwarded to node B and C. If the query is sent to node D, it is also limited within {D, E, F, G}. Therefore to broaden the search scope, it is necessary to integrate these two SIGs. We archive this as follows. Suppose node C and G both return node H's desired file. H asks C to make a DFS search in its SIG to find out whether node G is in it. If not, it means a SIG (that of G) disconnected with C is detected thereby C creates a connection with G, and then it asks one of its neighbors (*e.g.*, B) to link with one of G's neighbors (*e.g.*, F). In this way, the two SIGs are integrated.

4. Performance evaluation

4.1. Simulation methodology

The Internet node distance can be modeled as the geometric distance approximately [18], so we model the

underlay latency between two nodes as the geometric distance in a 3-dimensional Euclidean space. In general, Internet has the property that the nearer two nodes are located, the more likely they have a direct link. Waxman topology proposed in [27] is consistent with this property for the interconnection probability between two arbitrary nodes is

$$P(u,v) = \alpha e^{-d/\beta L}$$

where $0 \leq \alpha, \beta \leq 1$, d is the Euclidean distance between them, and L is the maximum distance between any two nodes. However, we require a 3-dimensional topology and Waxman only provides a 2-dimensional topology, so we set $z = \frac{x+y}{2}$, assuming z depends on x and y.

Three systems are compared: Gnutella, the CON-based system (referred to as *CON*) and the shortcut-based system [24] (referred to as *shortcut*). The simulation involves 2,500 nodes with average degree 4, and the TTL is set as 5. We deploy exact-match queries and do not use the above thresholds but simply *topic* and *id* to represent the file similarity, *i.e.*, the equivalence of topics means two files (or a file and a query) have similarity larger than T_s. The size of the topic space is 500 and the size of file id space for each topic is 50. Initially, each node is arbitrarily allotted five topics, each of which contains five files, randomly chosen out of its space. The nodes in CON initialize their neighbor sets based on links in its underlying network and for equality, the nodes in Gnutella and shortcut have the same number of neighbors but chosen randomly. We perform 2,500 queries each time, and repeat 20 times sequentially to represent the simulation length. To simulate users' bias in searching for files, desired files mainly belong to its local topics. For convenience, the local files are not retrieved so that the curves of Gnutella are relatively stable.

For simplicity, we do not order the table entries as mentioned above but just adopt the FIFO strategy. The table size is fixed at 5. The SIG is structured as undirected graph to broaden the search scope at some maintenance cost. The size of the shortcut list is set as 5 as well.

4.2. Metrics

We choose the following metrics to comprehensively measure the effectiveness of our method and compare it with other relevant systems.

- **Success rate:** The probability of finding the desired file before the search terminates. Remember our search is exact match so it is relatively low. It is one of the most critical metrics in designing Peer-to-Peer systems.
- **The average number of hops per query:** It shows how many hops on average are needed to complete a successful query.
- **The average latency per query:** Considering the topology mismatch, it is necessary to study the sum of latency along the search path to estimate the real delay of the query.
- **The average number of messages per node and the extra messages ratio:** It is used to measure the system overhead. The first one includes the cost to build CON as well as that for search, and the second is the percentage of messages to maintain file tables.
- **SIG rate:** As we mix the flooding-based search with the CON-directed search in CON, this metric describes the contribution of the file table in successful queries. Precisely, each step along the query path is tagged as whether it is forwarded based on SIG or flooding. Therefore it is possible to calculate the ratio of CON-directed steps to all steps.
- **Stretch:** The ratio of latency accumulated along the overlay path to that between the query initiator and file supplier. This metric is used to evaluate the effectiveness of our mechanism to alleviate topology mismatch problem.

4.3. Simulation results and analysis

Firstly, we examine the success rate, average latency and hops which reflect the search efficiency of the three systems.

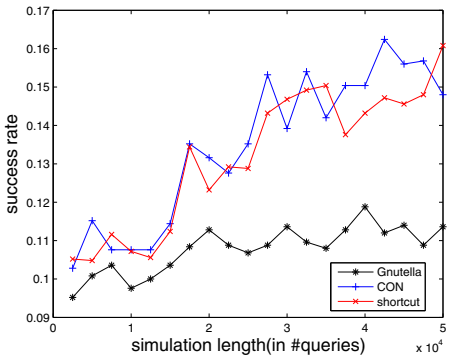

Figure 2. The success rate for queries.

The curves of Gnutella are rather smooth which proves our prohibition against local retrieval is really effective. Compared with Gnutella, CON performs the same at first but archives a substantial improvement as file tables are gradually filled. For the last 2,500 queries, the number of hops and latency in Fig. 4 and 3 are reduced to 35% and 31%, respectively. At the same time, Fig. 2 shows its success rate increases from 10% to 16%.

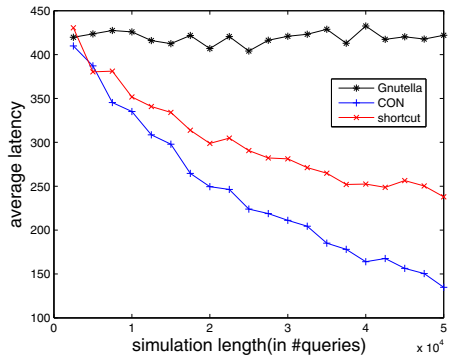

Figure 3. The average latency for queries.

CON also performs similarly with shortcut at first, and then with the same success rate, it outperforms shortcut in the number of hops and latency, *e.g.*, for the last 2,500 queries, the number of hops and latency are 75% and 56% of shortcut, respectively. The reduction of latency is more than that of hops, meaning that the mismatch problem is alleviated better in CON.

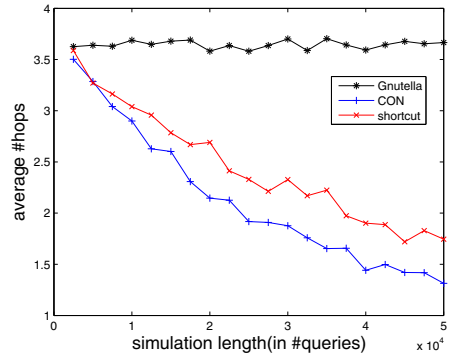

Figure 4. The average number of hops for queries.

Secondly, we study system overhead by calculating the average number of messages per node. Because of the different perceptions of *message* in CON and shortcut, we only compare CON as well as replicated CON (*i.e.*, CON with the square-root replication mentioned above) with Gnutella. we can see from Fig. 5 that the average number of messages per node is reduced by 41% in CON and 48% in replicated CON, respectively.

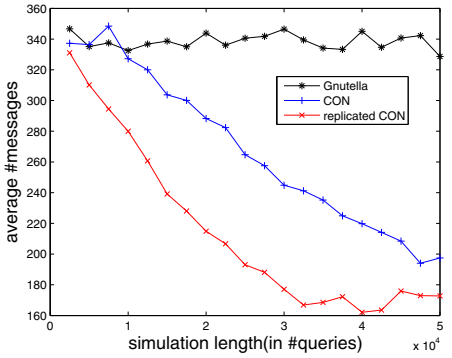

Figure 5. The average number of messages per node.

Thirdly, we study whether the file table works by examining the SIG rate for CON. Fig. 6 shows that as folders and tables are filled, the rate obviously increases from 5% to 88%, meaning that SIG plays an increasingly important role in search, and we conclude that the increase of query success rate is indeed archived by the CON-directed search rather than the flooding-based search.

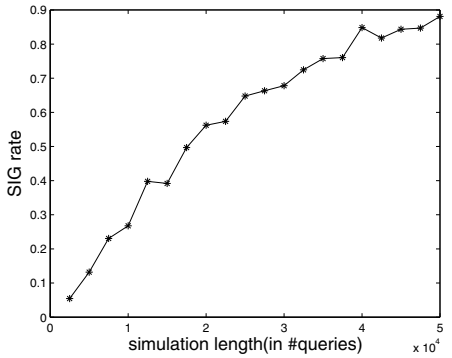

Figure 6. The SIG rate in CON, showing the utilization of the file table.

Fourthly, we study the extra messages ratio in the simulation. Fig. 7 plots that, because of our restriction on table size, this value is relatively small, increasing from 0.11% to 0.60%.

Fifthly, to examine whether the mismatch problem is alleviated in CON and compare it with that of the

other systems, the stretch is studied. In Fig. 8, CON reduces the stretch from 6.02 to 1.39, which is 24% of Gnutella and 53% of shortcut, respectively.

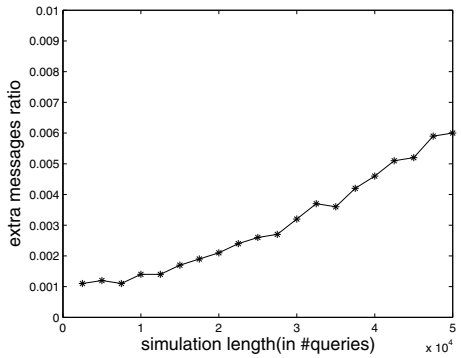

Figure 7. The extra messages ratio in CON.

Finally, the query success rate is studied while nodes continually crash and rejoin the system. We first randomly select 250 nodes (10% of all) and make them crash, then perform 5,000 queries. After that the 250 nodes rejoin the system, another 5,000 queries are performed. This procedure is repeated for ten times. Fig. 9 plots that CON performs better than Gnutella both when nodes crash and rejoin the system.

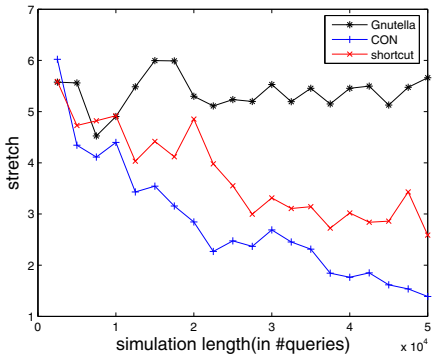

Figure 8. The stretch of Gnutella, CON and shortcut.

5. Further discussion

The method we choose to cluster nodes with similar interest is to utilize *content overlay network* to link them *logically*, which looks like the telephone conference. An alternative choice is to arrange files so that relevant files are clustered *geographically*. That is, by comparison, similar to the traditional conference, but the attendees are *files* rather than *nodes*. Tang et al. [26], to a certain extent, use this strategy but they are near in the node identifier space rather than its underlay proximity.

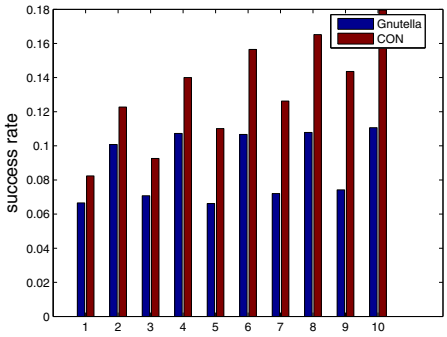

Figure 9. The success rate of CON and Gnutella when nodes continually crash and rejoin the system.

An interesting expanded utilization of *content overlay network* is based on the observation that traditional decentralized structured protocols unanimously arrange *nodes* according to the predefined structure such as ring, hypercube, etc. Though there is no doubt that such designs have their benefits, using *files* rather than *nodes* as the structure element has its own advantages. Let us take Chord [25] for example. The overlay network is constructed according to underlay proximity, but all *files* are organized as a Chord ring. When a new file is inserted, its identifier strictly defines its position in the ring. Its site in overlay network, on contrary, is arbitrary.

With the construction of ring, it has all the merits a structured protocol has, *e.g.*, the guarantee to find an existed file, the efficient route ($logN$ hops where N is the number of files) to locate it, etc. Additionally, if VSM is used as the file identifier (thus the files similar in semantics are near in the identifier space), it can also ensure the match between the content semantics and the underlying network, realizing the goal proposed in Paragraph I. More importantly, the freedom for file to select location facilitates the exploration of peer heterogeneity.

Similar with other structured protocols, the major drawback is that table entries contain other files' sites, costly to maintain when nodes continually join and leave the system. On one hand, we can use the constant degree protocol such as Koorde [11] to minimize the table size. On the other hand, piggyback should be implemented to avoid unnecessary update operation.

Because of space limitation, the detail is omitted.

6. Conclusion

Inefficient flooding-based search and topology mismatch are the major concerns in unstructured Peer-to-Peer system design. This paper proposes several techniques to alleviate these problems. Among them, the most significant contributions are 1) the use of *file* to form *node* special interest groups making the search more efficient, 2) the construction of the overlay network that well match its underlying network. Simulation results show that such enhancements make the query success rate higher, the response time shorter, the system bandwidth consumption smaller, and the topology match between the overlay network and its underlying network better.

7. Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by China NSF grants (60573131, 60673154), Jiangsu Provincial NSF grants (BK2005208, BG2007039), and China 973 project (2006CB303004). The Conference Participation is supported by Nokia Bridging the World Program.

References

[1] BitTorrent website. *http://www.bittorrent.com*.
[2] eDonkey website. *http://www.edonkey.com*.
[3] Gnutella website. *http://gnutella.wego.com*.
[4] Groove Virtual Office website. *http://www.groove.net*.
[5] KaZaA website. *http://www.kazaa.com*.
[6] Napster website. *http://www.napster.com*.
[7] SETI@home website. *setiathome.ssl.berkeley.edu*.
[8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like P2P systems scalable. *Proceedings of SIGCOMM 2003*, pages 407–418, 2003.
[9] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. *ICDCS 2002.*, pages 23–32, 2002.
[10] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. *Proceedings of SIGCOMM 2003*, 3, 2003.
[11] M. Kaashoek and D. Karger. Koorde: A simple degree-optimal distributed hash table. *Proceedings of the 2nd IPTPS*, pages 98–107, 2003.
[12] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. *Proceedings of the eleventh international CIKM*, pages 300–307, 2002.
[13] J. Liang, R. Kumar, and K. Ross. Understanding KaZaA. *available at http://cis.poly.edu/ross/papersUnderstandingKaZaA.pdf,2004*.
[14] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang. Location-Aware Topology Matching in P2P Systems. *Proceedings of IEEE INFOCOM*, pages 2220–2230, 2004.
[15] Y. Liu, L. Xiao, and L. Ni. Building a scalable bipartite P2P overlay network. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004.
[16] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. *Proceedings of the 16th ICS*, pages 84–95, 2002.
[17] W. Michael, D. Zlatko, and R. Elizabeth. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.
[18] T. Ng and H. Zhang. Towards global network positioning. *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*, pages 25–29, 2001.
[19] C. Plaxton, R. Rajaraman, and A. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed System. *Proceedings of the Symposium of Parallel Algorithms and Architectures*, pages 311–320, 1997.
[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. *A scalable content-addressable network*. 2001.
[21] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. *Proceedings of IEEE INFOCOM*, 3, 2002.
[22] M. Ripeanu and I. Foster. Mapping the Gnutella Network: Macroscopic Properties of Large-Scale Peer-to-Peer Systems. *First IPTPS*, 68, 2002.
[23] S. Saroiu, P. Gummadi, S. Gribble, et al. A measurement study of peer-to-peer file sharing systems. *Proceedings of MMCN*, 2002.
[24] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. *Proceedings of IEEE INFOCOM*, 2003.
[25] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of SIGCOMM 2001*, 31(4):149–160, 2001.
[26] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. *Proceedings of SIGCOMM 2003*, pages 175–186, 2003.
[27] B. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.
[28] L. Xiao, Y. Liu, and L. Ni. Improving unstructured peer-to-peer systems by adaptive connection establishment. *Computers, IEEE Transactions on*, 54(9):1091–1103, 2005.