Dominating-Set-Based Searching in Peer-to-Peer Networks ⋆

Chunlin Yang[1] and Jie Wu[2]

[1] Siemens Network Convergence LLC, Boca Raton, FL 33487
[2] Dept. of Comp. Science & Eng., Florida Atlantic University, Boca Raton, FL 33431

Abstract. The peer-to-peer network for sharing information and data through direct exchange has emerged rapidly in recent years. The searching problem is a basic issue that addresses the question "Where is X". Breadth-first search, the basic searching mechanism used in Gnutella networks [4], floods the networks to maximize the return results. Depth-first search used in Freenet [3] retrieves popular files faster than other files but on average the return results are not maximized. Other searching algorithms used in peer-to-peer networks, such as iterative deepening [9], local indices [9], routing indices [2] and NEVRLATE [1] provide different improved searching mechanisms. In this paper, we propose a dominating-set-based peer-to-peer searching algorithm to maximize the return of searching results while keeping a low cost for both searching and creating/maintaining the connected-dominating-set (CDS) of the peer-to-peer network. This approach is based on random walk. However, the searching space is restricted to dominating nodes. Simulation has been done and results are compared with the one using regular random walk.

1 Introduction

Peer-to-peer network models such as Gnutella [4], Freenet [3], and Napster [5] are becoming popular for sharing information and data through direct exchange. These models offer the important advantages of decentralization by distributing the storage capacity and load across a network of peers and scalability by enabling direct and real-time communication. In fully decentralized peer-to-peer networks, there is no need for a central coordinator. Communication is individually handled by each peer. This has the added benefit of eliminating a possible bottleneck in terms of scalability or reliability. The peer-to-peer approach offers an alternative to traditional client-server systems for some application domains. It circumvents many problems of client-server systems but results in considerably more complex searching, node organization and reorganization, security, and so on [7].

The searching problem is a basic issue that addresses the question "Where is X". Breadth-first search (BFS), the basic searching mechanism used in Gnutella

⋆ This work was supported in part by NSF grants CCR 0329741, ANI 0073736, and EIA 0130806. Contact person: jie@cse.fau.edu.

networks [4], floods the networks to maximize the return results. Depth-first search (DFS) used in Freenet [3] retrieves popular files faster than other files but on average the return results are not maximized. Other searching algorithms used in peer-to-peer networks, such as iterative deepening [9], local indices [9], routing indices [2] and NEVRLATE [1] provide different improved searching mechanisms. In this paper, we propose a dominating-set-based peer-to-peer searching algorithm to maximize the return of searching results while keeping a low cost for both searching and creating/maintaining the connected-dominating-set (CDS) of the peer-to-peer network. A connected-dominating-set (CDS) [8] of a peer-to-peer network is a connected subset of nodes of the network from which all nodes in the network can be reached in one-hop. Finding a minimum CDS is NP-complete for most graphs. Wu and Li [8]'s marking process gives a simple and distributed algorithm for calculating CDS. In this paper, we propose a peer-to-peer network searching algorithm using CDS generated by the marking process with some modification of the reduction rules 1 and 2 to maximize the searching results while keeping the cost of searching and maintaining the CDS low. This approach is based on random walk. However, the searching space is restricted to dominating nodes. Simulation results have been presented and discussed.

2 Related Work

Gnutella [4] is the foremost large-scale, fully decentralized directory and distribution system running on the Internet. It uses a BFS with predefined depth D, where D is the system-wide maximum time-to-live (TTL) of a message in hops. Upon receiving a request, a node sends a query to all its neighbors and each neighbor searches its own resources and forwards the message to all of its own neighbors. If a query is satisfied, a response will be sent back to the original requester using the reverse path. Queries are assigned unique IDs to avoid repetition. Gnutella uses a TTL of 7 (about 10000 nodes) to avoid network congestion [6]. BFS can still be cyclical, and can cause excessive traffic and waste resources.

In Freenet [3], information is stored on hosts under searchable keys. It uses a DFS with depth limit D. Each node forwards the query to a single neighbor, and waits for a definite response from that neighbor. If the query was not satisfied, the neighbor forwards the query to another neighbor. If the query was satisfied, the response will be sent back to the query source using the reverse path. Each node along the path copies data to its own database as well. In this approach, more popular information becomes easier to access. However, DFS suffers from poor response time.

Iterative Deepening [9] searching method initiates multiple DFSs with successively larger depth limits, until the query is satisfied or the maximum depth has been reached. Searching in Local Indices [9] needs to store information of all nodes within a defined number of hops. Each node maintains an index of the data of all nodes within r hops, where r is a system-wide variable known as the radius of the index. When receiving a query, a node can process it on behalf of every node within r hops, data can be searched on fewer nodes to reduce the cost while keeping the query satisfaction.

In NEVRLATE [1] (Network-Efficient Vast Resource Lookup At The Edge), directory servers are organized into a logical 2-dimensional grid, or a set of servers enabling registration (publish) in one "horizontal" dimension and lookup in the other "vertical" dimension. Each node is a directory server. Each set of servers, the vertical cloud, can reach each other member of the set. The set of sets of servers is the entire NEVRLATE network. Each host registers its resource and location to one node of each set. When a query comes, only one set needs to be searched to get all locations containing the satisfied query information.

Routing Indices [2] uses only single DFS but allows a node to select the "best" neighbor to send a query to. Routing Indices is a data structure and associated algorithms that, given a query, returns a list of neighbors ranked according to their *goodness* for the query. The goodness is measured by k-hop ranking which is a weighted total number of documentations within k hops. In general, the larger the k the smaller the weight assigned to the number of documentations at k hops. The 0-hop ranking reflects the number of documentations associated with the node. However, if u has a higher k-hop ranking than v, it does not imply that u has a higher 0-hop ranking than v.

3 Extended Marking Process

The dominating-set-based searching algorithm defined below tries to maximize the return results while minimizing the searching and maintenance costs. No global information is needed to construct and reduce the CDS using the marking process and reduction rule 1 and rule 2 [8].

Specifically, the marking process is a localized algorithm described in [8] in which hosts interact only with others in a restricted vicinity. Each host performs exceedingly simple tasks such as maintaining and propagating information markers. Collectively, these hosts achieve a desired global objective of finding a small CDS. The marking process marks every vertex in a given connected and simple graph $G = (V, E)$. $m(v)$ is a marker for vertex $v \subset V$, which is either T (marked) or F (unmarked). The marking process consists of: (1) Initially, assign marker F to each v in V. (2) Each v exchanges its open neighbor set $N(v)$ with all its neighbors. (3) Each v assigns its marker $m(v)$ to T if there exist two unconnected neighbors. It is shown that given a graph $G = (V, E)$ that is connected but not completely connected, the vertex subset V', derived from the marking process, forms a connected dominating set of G.

Two localized reduction rules 1 and 2 [8] are provided to reduce the size of the CDS: if the neighbor set of node u in the CDS is covered by that of another node v or those of two connected nodes v and w in the CDS, then node u can be removed from the CDS. In this case, u is said to be covered by v (or by v and w). To avoid simultaneous removal of two nodes covering each other, each node u is assigned a distinct id. A node is removed from the CDS when it is covered by node(s) with higher id(s).

In this paper, we modify rules 1 and 2 [8] to use the a special 1-hop ranking, denoted as *docs*, of each node as the priority to break a tie. 1-hop ranking, *docs*, is defined as *the total documentation number of node v plus the highest documentation number of a v's neighbor.* We treat all types of documentation the same and will classify them in the future research. To get a unique total number of documentations, we can easily assign a unique node id. In case of a tie in the 1-hop ranking, node id is used to break a tie. Here are the modified rules 1 and 2:

Rule 1: Consider two vertices v and u in V'. If $N(v) \subset N(u)$ in G and $docs(v) < docs(u)$, change the marker of v to F if node v is marked, i.e., G' is changed to $V' - v$.

Rule 2: Assume that u and w are two marked neighbors of marked vertex v in V'. If $N(v) \subset N(u) \cup N(w)$ in G and $docs(v) = min(docs(v), docs(u), docs(w))$, then change the marker of v to F.

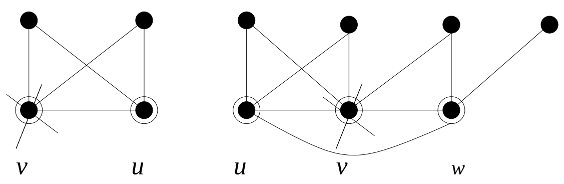

Fig. 1. Examples of rule 1 (left) and rule 2 (right).

In the example shown in Figure 1 (left), using the marking process and above modified reduction rule 1, node u will be the only dominating node in the graph if node u has a higher 1-hop ranking than node v. If node u has a lower 1-hop ranking than node v, node v will be the only dominating node in the graph. In the example shown in Figure 1 (right), node v can be eliminated from the dominating set based on rule 2 if node v has the minimum *docs* within these three nodes u, v, and w.

4 Dominating-Set-Based Peer-To-Peer Searching Algorithm

For a peer-to-peer network, we can use the above marking process with the modified rules 1 and 2 to get a CDS and use this subset of nodes for the searching in the network. The searching process resembles a random walk with an assigned TTL (depth). The searching process stops when TTL expires or a visited node is reached again. Unlike regular random walk, the searching process is restricted to dominating nodes only. In addition, we allow "one-hop branches" along the walk when certain conditions are met. Each one-hop branch connects to a non-dominating neighbor with the maximum 0-hop ranking in the neighborhood of the corresponding dominating neighbor.

1. First calculate 0-hop ranking and 1-hop ranking (*docs*) of each node.
2. Use the above defined marking process and modified reduction rules 1 and 2 to get a CDS.
3. When a node S receives a request, S searches from its own database and returns the results to the requester if there is any documentation.
4. If node S is the original request node and is not a dominating node (marked as F), it forwards the request to the dominating neighbor (marked as T) which has the highest 1-hop ranking among all of its dominating neighbors. If node S is not the original requestor nor a dominating node, it will not send a query to any of its neighbors.
5. If node S is a dominating node, it sends the request to the dominating neighbor with the highest 1-hop ranking among all of its dominating neighbors. Node S also sends the request to the non-dominating neighbor which has the highest 0-hop ranking among all of its neighbors (dominating neighbors and non-dominating neighbors) if there is one.
6. Repeat steps 3 to 5 until either the maximum number of hops is reached or a visited node is reached again.

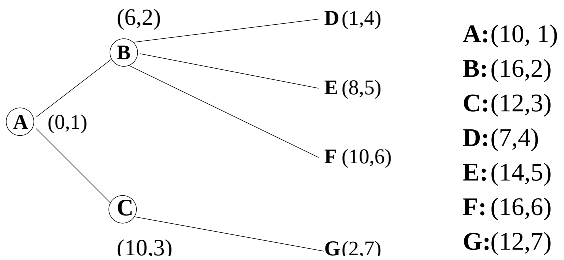

Fig. 2. A sample network with three dominating nodes.

For example, in Figure 2, each node has two numbers: the first number is the 0-hop ranking of that node and the second number is the unique node id. In Figure 2, the right column lists *docs* (i.e. 1-hop ranking) of each node. When node A gets a request, it first searches its own database. Then node A sends the request to node B which is the dominating neighbor with a higher *docs* than node C. Node B sends the request to node F, which has the highest priority (0-hop ranking and node id) of all neighbors of node B, and gets additional 10 documentations. Overall, this searching process gets a total of 16 documentations. If node F is a requester, it will send a message to node B and then to A and C to get a total of 26 documentations. Note that in this case G will not be searched, since it has a smaller 0-hop ranking than the dominating node C.

Notice if in the same example we use random walk by treating all nodes the same and by sending a message to the “best” ranking neighbor, node A will send the request to node B, which has a higher 1-hop ranking than the other neighbor C, node B will send the request to node F. If node F is the requester, after sending the message to node B, node B will choose node E for the next searching because E has the next highest 0-hop ranking in all non-dominating neighbors of B. The process will stops because E has no more unvisited neighbor. The the searching process initiated from F will get 18 documentations instead of 26 as in the dominating-set-based approach. The searching stops before the defined maximum TTL (we use 20 in our simulation) has been reached.

The above proposed dominating-set-based searching algorithm in peer-to-peer networks needs only local information to get the CDS. Wu [8] has proven that the process calculates CDS in $O(\Delta^2)$ time with distance-2 neighborhood information, where Δ is the maximum node degree in the network. The above modified reduction rules 1 and 2 will not change the complexity of the calculation of the CDS and reduction. The proposed algorithm also uses constant (2) rounds of message exchange.

In general, a ranking just gives a "direction" towards the document, rather than its actual location. To create and maintain an "accurate" routing ranking, global information is needed or we can use *hop count ranking* [2] which also needs information from nodes within a certain number of hops. More cost will be added to create and maintain ranking of all affected nodes of the network if there are loops in the network.

In the CDS approach, when a node is added or dropped from the network, only its neighbors will be notified whereas the routing indices will update almost all nodes within k-hops to keep k-hop ranking up-to-date. In our simulation, we will see that the average ranking of all dominating nodes will be higher than the non-dominating nodes. That means by restricting the searching space to dominating nodes more documentations can be found.

5 Simulation

A C++ program has been implemented to simulate a peer-to-peer network searching using random walk and CDS described above. The network is randomly generated with a maximum degree of max_degree, each node is randomly assigned a number from 0 to max_docs documentations. To generate the network, 1 or 2 nodes with a degree less than max_degree are randomly selected from the connected set, which was initially assigned with one node, to connect to a new selected node from the unconnected set. This procedure repeats until all nodes in the unconnected set get connected to the connected set. This will guarantee that the result network is connected and can have loops. After all nodes get connected and assigned a number of documentations, both 0-hop ranking and 1-hop ranking are calculated for each node. The above defined marking process with modified rules 1 and 2 will mark/unmark a node to a T (a dominating node) or F (a non-dominating node) to get a CDS for the network.

Table 1 shows the average 1-hop ranking for all nodes, dominating nodes and non-dominating nodes in a network with 50,000 nodes. It is clear that dominating nodes have a higher average 1-hop ranking than non-dominating nodes. This is

Max degree	All nodes	DS nodes	Non-DS nodes
2	2.5230	2.6735	1.9856
5	6.1677	6.4964	4.9937
10	12.1713	12.7712	10.0288

Table 1. Average 1-hop ranking in a network with 50,000 nodes.

due to the fact that the dominating nodes have higher connectivity than non-dominating nodes. Based on the modified rules 1 and 2, the dominating nodes have more $docs'$ than non-dominating nodes.

Two searching methods are simulated. First, in regular random walk, a request will only be sent to the "best" neighbor with the highest 1-hop ranking. The second searching method is dominating-set-based, which sends a request to the "best" dominating neighbor which has the highest 1-hop ranking of all of its dominating neighbors and to the non-dominating neighbor which has the highest 0-hop ranking if there is one. We used 20 hops as the maximum number for both searching methods.

In Table 2, *search_no* is the number of trials. RW_docs (RW_cost) is average documentation returned from (average number of hops using) random walk searching. DS_docs (DS_cost) is average documentation returned from (average number of hops using) dominating-set-based searching. We can see that the number of documentations retrieved from dominating-set-based searching is much higher than the number returned from random walk searching. In DS_cost, two costs are recorded, $c_1(c_2)$, where c_1 is the number of steps in depth and c_2 is the number of one-hop branches. RW_avg (DS_avg) measures the average number of documentations returned for each step (in depth) in random walk searching (dominating-set-based searching). In general, DS_avg is higher than RW_avg. The reason for this is that dominating-set-based searching sends requests to the "best" dominating neighbor which has higher connectivity and higher *docs* and to the non-dominating neighbor which has the maximum number of documentations. As mentioned before, the cost of maintaining CDS in dominating-set-based searching is minimum since only local information is required.

Based on the results of Tables 1 and 2, we have the following conclusions: (a) Dominating-set-based searching terminates later than random walk searching. As a result, the average number of documentations returned from dominating-set-based search is more than random walk searching. (b) The number of "one-hop branches" is relatively insignificant. This is because such branches are generated only when it has a higher 0-hop ranking than its dominating node. (c) In dominating-set-based searching, the average number of documentations returned per step (in depth) is higher than random walk searching.

6 Conclusion

In this paper we have proposed a peer-to-peer searching algorithm using CDS. The CDS is constructed using the marking process [8] with the modified rules 1

nodes_no	5,000	5,000	5,000	50,000	50,000	50,000
max_docs	2	5	10	2	5	10
max_degree	6	6	6	6	6	6
search_no	100	100	100	100	100	
RW_docs	19.88	40.02	102.78	15.35	42.86	92.83
RW_cost	13.00	10.74	14.25	10.04	11.21	12.53
RW_avg	1.5292	3.7262	7.2126	1.5288	3.8233	7.4086
DS_docs	30.63	75.71	148.66	33.01	80.11	161.47
DS_cost	17.92 (2.37)	18.03 (2.92)	18.62 (2.64)	18.60 (3.27)	18.29 (3.59)	19.31 (2.54)
DS_avg	1.7103	4.1991	7.9838	1.7747	4.3799	8.3619

Table 2. Searching results.

and 2 for reduction. Simulation shows that dominating-set-based searching returned more documentations than random walk searching and kept the searching cost low. The cost of creating and maintaining the CDS is lower than that of the cost to create and maintain the routing indices ranking or hop count ranking as in [9]. Our future research will focus more on in depth simulation of dominating-set-based approach using different searching algorithms, such as DFS. Other ways of defining k-hop ranking ($k > 1$) will also be explored.

References

1. A. Chander, S. Dawson, P. Lincoln, and D. Stringer-Calvert. Nevrlate: scalable resource discovery. *Proc. 2nd IEEE/ACM Int'l Symposium on Cluster Computing and the Grid*, pages 382–388, 2002.
2. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. *Proc. 22nd Int'l Conference on Distributed Computing Systems*, pages 23–32, 2002.
3. http://freenet.sourceforge.net (Freenet website).
4. http://www.gnutella.com (Gnutella website).
5. http://www.napster.com (Napater website).
6. M. Portmann and A. Seneviratne. The cost of application-level broadcast in a fully decentralized peer-to-peer network. *Proc. 7th Int'l Symposium on Computers and Communications*, pages 941–946, 2002.
7. M. Krishna Ramanathan, V. Kalogeraki, and J. Pruyne. Finding good peers in peer-to-peer networks. *Proc. Int'l Symposium on Parallel and Distributed Processing*, pages 232–239, 2002.
8. J. Wu and H Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. *Proc. 3rd Int'l Workshop on Discrete Algorithm and Methods for Mobile Computing and Communications*, pages 7–14, 1999.
9. B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. *Proc. 22nd Int'l Conference on Distributed Computing Systems*, pages 5–14, 2002.