Scalable Ultrapeer-based DHT Protocol for File Sharing

Chayoung Kim and Jinho Ahn
Dept. of Computer Science, Kyonggi University, Suwon-si Gyeonggi-do, Korea
kimcha0@kgu.ac.kr, jhahn@kyonggi.ac.kr

ABSTRACT

In this paper, we present a novel file searching protocol to structure a DHT ring consisting of only ultrapeers, not all the nodes. The DHT ring in this protocol is much less sensitive to the churn rate because ultrapeers have much longer uptime compared with leaf nodes. Thus, this feature makes the protocol more scalable and efficient than the previous DHT ones in terms of costs of file search, node join and leave operations and the number of routing table entries each node should maintain. Moreover, it is more effective for locating rare files than Gnutella-like searching protocols.

KEYWORDS: P2P system, file sharing, ultrapeer, dynamic hash table, scalability

1. INTRODUCTION

Recently, peer-to-peer (P2P) architectures have been gaining extremely high popularity for file sharing applications instead of centralized and client-server models [14]. The peer-to-peer file sharing started with the introduction of Napster [21]. Napster achieves a highly functional hybrid P2P design by using centralized directory search facility while file downloading function is decentralized. In this system, real peers possessing sharable contents register with centralized directory servers. These servers store reference addresses and transfer rates of peers into their databases for looking up location of desired contents. However, though directory servers don't handle directly these downloading services, these servers manage a lot of shared files' lists. Thus, centralized directory servers cannot scale up well in large-scale P2P systems because of requiring extremely expensive and unmanageable bandwidth or server farms and overloading the servers due to very large network traffic resulting from the centralized search functionality. Moreover, if the servers crash, peers cannot share and search files any longer.

To solve the scalability of content search and the single point of failure, decentralized P2P systems such as Gnutella [22] can be used. In Gnutella, peers transmit to all its neighbors query messages requesting files by flooding. In this case, Gnutella uses a Time to Live (TTL) to restrict the scope of the flooding. But this random search method is not inherently scalable [13]. So, the initial version of Gnutella does not scale up well because of the bandwidth overwhelmed by broadcast messages and the computing cycles consumed by many peers [3]. To deal with this problem, hierarchical Gnutella network was proposed by categorizing nodes on the network into leaf nodes and ultrapeer nodes [19, 20, 11]. An ultrapeer acts as a searching proxy for the Gnutella network by having the leaf nodes register with it. The ultrapeer takes charge of much high load on it because it has better networking capability and CPU power than leaf nodes. So, Gnutella has obtained an important advantage of significant reduction of the flooding overhead. Also, this system includes dynamic routing method [8]. The search model of the system adjusts TTL of outgoing queries dynamically. This method initially decreases TTL of each outgoing queries and broadcasts the queries to neighboring peers. When a desired number of results are received, the search phase is finished. Otherwise, dynamic querying method increases the value of TTL of its queries and floods the query messages until the corresponding user gets enough query results from the network.

Distributed Hash Table (DHT) is typically used for building distributed lookup services of structured P2P systems [3, 4, 6, 10, 16, 18, 9, 15]. DHT is implemented by organizing N peers in a structured overlay P2P system. It maintains a table of routing entries of a small number of neighboring peers and requires O(logN) peer hop counts in searching a particular file [9]. But P2P clients may be extremely transient and do not have equal bandwidth capabilities or computational power or other properties (like uptime, connection capability, etc.). The DHT systems are likely to be less resilient in the face of transient user population whereas unstructured overlay network systems are relatively less sensitive to this churn

978-1-4244-4586-8/09/$25.00 ©2009 IEEE

rate [5]. Second, structured designs are well-suited for exact-match queries and effective for looking up rare files. However, these DHT structured systems may not commonly be used compared with Gnutella-like systems because keyword searching is more prevalent and general than exact-match queries [13].

There are several modifications to Gnutella's design in order to accommodate the natural heterogeneity present in most peer-to-peer systems [5]. In this design, more powerful nodes that have higher bandwidth connectivity and uptime should be high degree nodes. These nodes will have pointers to a large number of files adjacent peers keep. With this topology, some adaptation algorithms may be used to ensure congruence between high capacity nodes and high degree nodes and guarantee that high capacity nodes are capable of providing answers to a greater number of queries by one-hop replication. Also, high capacity nodes are able to answer the queries much more efficiently by some search protocols based on a biased random walk model.

But, Gnutella and other similar systems support popular keywords searching because of assumptions that most searches are for hay, not needles [5]. So, these systems can be expensive to search for rare files and, in some cases, may not find such files. In this paper, we present a DHT-based file searching protocol to structure a DHT ring consisting of only ultrapeers, not all the nodes. Most of recently developed DHT-based searching protocols construct an application-level overlay network consisting of N nodes, where N is the total number of peers in the system. However, in this paper, only ultrapeers in a Gnutella-like network composes a DHT-based ring topology. The DHT ring in this protocol is much less sensitive to the churn rate because ultrapeers have much longer uptime compared with leaf nodes. Thus, this feature makes the protocol more scalable and efficient than the previous DHT ones in terms of costs of file search, node join and leave operations and the number of routing table entries each node should maintain. Moreover, it is more effective for locating rare files than Gnutella-like searching protocols

2. THE PROPOSED PROTOCOL

2.1. DHT Ring Construction and File Searching

In Gnutella, nodes having powerful CPU and connection capability become ultrapeers and serve as network hubs that have pointers for indexed files belonging to leaf nodes. Ultrapeers maintain many leaves' connections and leaf nodes maintain only a single connection to an ultrapeer. In distributed hash tables(DHTs), given the exact name of a file, the location of the file can be found by exact match lookups.

In this paper, a DHT structured overlay network is organized by only ultrapeers, not all peers. In figure 1, there is the DHT ring composed of four ultrapeers without leaf nodes. This feature will make the protocol efficient in terms of the cost of file search than the other DHT ones[7,17] .

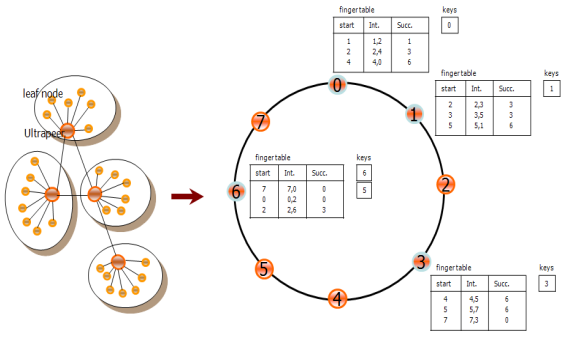

Figure 1. The DHT Ring Composed Of Four Ultrapeers In The Proposed Protocol

In figure 2, there is a file search algorithm. For example, to find a file "id", a peer requests an ultrapeer "u" to find a successor of "id". If there is "id", then it returns "id". Otherwise, it searches for the closest predecessor of "id" in the finger table. If there is the closest predecessor in finger table, then it returns the key of the ultrapeer.

```
// request ultrapeer u to find id's successor
u.find_successor(id)
 if (id ∈ (u, successor))
       return successor
 else
       u` = closest_preceding_ultra(id);
       return u`.find_successor(id);

//search for the closest predecessor in the local table of id
n.closest_preceding_node(id)
 for i = m downto 1
       if (finger[i] ∈ (u, id))
             return finger[i];
 return u
```

Figure 2. File Search Algorithm

2.2. Node Join and Leave

2.2.1. Leaf

In dynamic networks, nodes join and leave at any time. In most of previous DHT ones, each node maintains it's successor node, and the successor, denoted by

successor(k), is responsible for key k whenever nodes join and leave. For example, when a node joins the network, the node's identifier is chosen by hashing(like SHA-1) the node's IP address[18]. The key k m-bit identifier is ordered in an identifier circle modulo 2^m. In this paper, the proposed DHT system is structured by only ultrapeers without leaf nodes. Ultrapeers periodically send queries for content indexing to their connected leaf nodes, which reply with indexes of their respective shared files. So, ultrapeers use these replies to build and maintain an index directory of their leaves' contents.

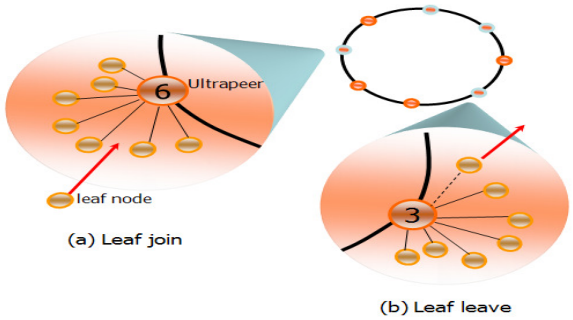

Figure 3. (a) Leaf Join (b) Leaf Leave

We can also see that a leaf node sends ultrapeers file sharing information when it joins the network and an ultrapeer updates its index directory about information of newly joined leaf node without restructuring the DHT structure in figure 3(a). In figure 3(b), when a leaf node leaves the network, an ultrapeer eliminates the information of the leaf or moves its shared files to another adjacent leaf.

```
// Leaf l join in existing ultrapeer u`
l.join(u`)
    send_file_index(u`, file_index_data);

// Leaf l Leave
l.leave(u`)
    if (l has any file)
        l` = choose_leaf(u`)
        move_files(l`);
        disconnect(u`);
```

Figure 4. Leaf Join and Leave Algorithm

There is a leaf join and leave algorithm in figure 4. When leaf l joins the group of existing ultrapeer u', leaf 1 sends index information of its shared files to ultrapeer u'. Then, the ultrapeer updates its DB using this information. When leaf 1 with any files leaves, ultrapeer u' selects another leaf 1' and leaf l uploads its shared files to leaf 1'. After that, leaf 1 disconnects with ultrapeer u'.

2.2.2. Ultrapeer

There is an assumption that ultrapeers seldom repeat to rejoin and should have much longer connection to Gnutella network. So, in this paper, we propose the protocol to have an effect of significant reduction on restructuring the DHT. This feature is different from those of the previous DHT protocols when nodes join and leave.

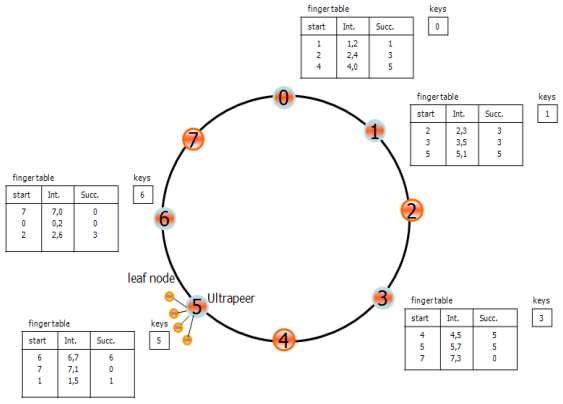

Figure 5. Ultrapeer Join

Figure 5 shows ultrapeer 5 joins newly in identifier circle consisting of 4 ultrapeers in the DHT. In the proposed protocol, each ultrapeer maintains a predecessor pointer. An ultrapeer's predecessor pointer contains the identifier and IP address of the immediate predecessor of the ultrapeer. Our protocol initializes the predecessor and fingers of ultrapeer n, updates the fingers and predecessors of existing ultrapeers to reflect the addition of new ultrapeer n, and transfers the ultrapeer n the state associated with keys that it has to be responsible for. In this example of figure 5, the keys that ultrapeer 6 has taken charge of are moved to ultrapeer 5 for considering load balancing.

When a node n leaves in previous DHT ones, the keys for which n is responsible are transferred to an neighboring one in the counter clockwise direction on the identifier circle[18]. But in the proposed protocol, ultrapeers need not be performed in most cases like those nodes in the existing ones because our DHT ring network is organized by only ultrapeers without leaf nodes. When an ultrapeer intends to leave from the network, any ultrapeer-capable leaf node directly connecting to the leaving ultrapeer can become a new ultrapeer. If there is no leaf node that is eligible to be an ultrapeer, new connections should be established from the leaf nodes to another ultrapeer on the identifier circle.

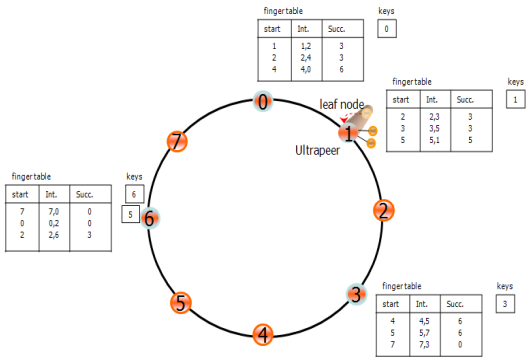

Figure 6. One Leaf Replacing Its Ultrapeer

Figure 6 illustrates there are some ultrapeer-capable leaf nodes covered by an ultrapeer when it leaves from the DHT network. In this case, a new ultrapeer is elected among them and takes over ultrapeer 1's role. At this time, this procedure doesn't require any additional changes for restructuring the DHT network.

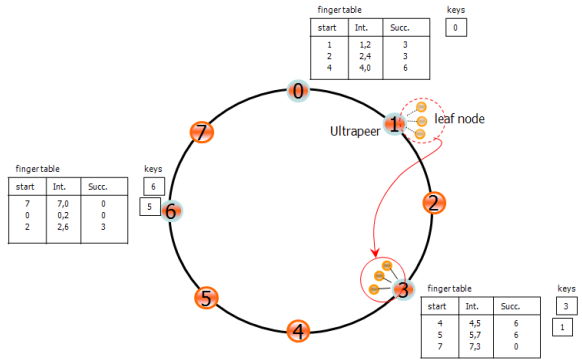

Figure 7. No Leaf Replacing Its Ultrapeer

Figure 7 shows that there is no ultrapeer-capable leaf node as a substitute for ultrapeer 1 as the ultrapeer attempts to leave. In this case, its covering leaf nodes should be administered by its adjacent ultrapeer, ultrapeer 3, and its directory information, transferred to ultrapeer 3.

3. PERFORMANCE COMPARISONS

In this section, we compare performance of our proposed ultrapeer-based DHT protocol with that of the recently representative one such as Chord[18]. In this comparison, the total number of the nodes in a network is N. A node in Chord requires O(logN) other nodes to resolve a lookup and $O(\log^2 N)$ messages to update routing information on node join and leave, and maintains a finger table with (logN) entries. In our protocol, we need O(logn) other nodes for efficient routing, $O(\log^2 n)$ messages for re-establishing routing structure and (logn) finger table entries in an N-node network with ultrapeers *n* nodes, respectively. For example, if N=1024 in a network with 16 ultrapeers, a Chord node requires 10 times of forwarding until the corresponding query is delivered to the closest node. But, in our protocol, an ultrapeer can find a desired file within at most 4 times. Therefore, we can see that our ultrapeer-based DHT is much more scalable and efficient than Chord-like DHT ones.

Figure 8 shows the maximum number of hop count for file searching as the number of nodes increases to join a network. We can see that the maximum number of hop count of the proposed protocol is less than that of Chord in performing lookups. Also, if the more ultrapeers have better networking capabilities and CPU power, each ultrapeer can accept much more leaf nodes and the number of hop count may decreases. Therefore, the proposed protocol can significantly reduce the searching time of resolving queries.

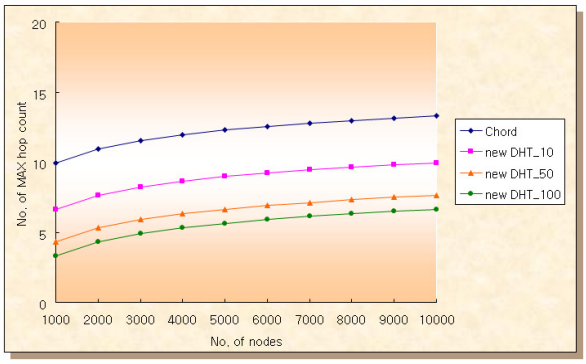

Figure 8. No. Of Nodes Vs. No. Of Maximum Hop Count

4. RELATED WORKS

The earlier P2P file sharing systems such as Napster[21] relies on a centralized directory to locate files. While this was sufficient for the early days of P2P, it has inherent reliability and scalability problems that make it vulnerable to attacks on the database. Some systems like Gnutella[22] and Kazaa[23], at the other end of the spectrum, are for the consumer to broadcast a query

message to all its neighbors. These systems form an unstructured overlay network in which each P2P node connects to several other nodes. The overlay topology is ad hoc and the placement of data is completely unrelated to the overlay topology. Searching on such networks is generally based on random walks, where various nodes may arbitrarily be probed and asked if they have any files matching a particular query. Pastry[7], Chord[18] and CAN[17] represent a second generation of P2P routing and location schemes that were inspired by the pioneering work of systems like Gnutella. Unlike that earlier work, they guarantee a definite answer to a query in a bounded number of network hops, while retaining the scalability and the self-organizing properties. In Pastry[7], nodes are responsible for the keys that are the closest numerically with the key space considered as a circle. The $|L|$ close nodes as neighbors compose a leaf set L. Routing is performed by forwarding the query to the neighboring node that has the longest shared prefix with the key. Chord[18] also uses a one-dimensional circular key space. The node responsible for the key is the key's successor whose identifier most closely follows the key. Also, each node has a successor list of k nodes that immediately follow it in the key space. CAN[17] chooses its keys from a *d*-dimensional toroidal space. Each node is associated with a hypercubal region of this key space, and its neighbors are the nodes that own the contiguous hypercubes. Query routing is forwarding each query to a neighbor that is closer to the key. However, these structured designs are likely to be less resilient at a high churn rate.

There are several attempts to improve the search facilities in Gnutella. Adamic et al.[1] takes power-law random graphs as a given and instead asks how to search on them best. They suggest the use of random walks, but that these random walks should be biased to seek out high-degree nodes. However, their work does not take into account the query load on individual nodes. Krishmanurthy et al.[12] propose a cluster-based architecture for P2P systems, which uses a network-aware clustering technique based on a central clustering server to group hosts into clusters. Each cluster has one or more delegate nodes that act as directory servers for the objects stored at nodes within the same cluster. In some sense, the high capability nodes in GIA[5] provide functionality similar to that of delegate nodes.

5. CONCLUSION

We present a novel file searching protocol to structure a DHT ring consisting of only ultrapeers, not all nodes, for locating rare files with low cost file search and considerably improving speed and accuracy of query matches than Gnutella-like searching ones. In an N-node network, a node of the previous DHTs requires routing information about (logN) other nodes for searching the key of a desired file and $O(\log^2 N)$ messages to update routing information in the network on node join and leave operations. But in our protocol, if the number of ultrapeers is *n* in the N-node network, the protocol needs only O(logn) routing information and $O(\log^2 n)$ maintenance messages respectively. This feature allows our protocol to be much more scalable and resilient to the churn rate than the existing DHT ones.

For future work, we attempt to apply the proposed protocol to our user-friendly Gnutella system [2] previously developed and measure its performance gains in various aspects.

ACKNOWLEDGEMENT

This work was supported by Gyeonggido Regional Research Center program grant (Development and Industrialization of Integrated Frameworks for Very Large-scale RFID Services) [Corresponding Author: Jinho Ahn].

REFERENCES

[1] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman,"Search in power law networks", Physical Review. E Vol. 64, pp. 46135-46143, 2001.

[2] S. Bang and J. Ahn, "Development of effective P2P systems for file sharing", Journal of Kyonggi University Basic Science Research Institute, Vol. 19, No. 1, 2006.

[3] H. Balakrishnan, F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems", Communications of the ACM, Vol. 46, No. 2, pp. 43-48, Feb. 2003.

[4] A. Clements, D. Ports, and D. Karger, "Arpeggio: Metadata Searching and Content Sharing with Chord", 4th International Workshop on Peer-To-Peer Systems, LNCS vol. 3640, pp.58-68, Ithaca, New York, Feb. 2005.

[5] Y. Chawathe, S. Ratnasamy, L. Breslay, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable", In ACM SIGCOMM, Germany, IRB-TR-03-014, Aug. 2003.

[6] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with Chord, a distributed location service", In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII), pp. 81-86, May 2001.

[7] P. Druschel and A. Rowstron “Pastry:Scalable, distributed object location and routing for large-scale peer-to-peer systems”, In Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001), pp. 329-350, Nov. 2001.

[8] A. Fisk, "Gnutella dynamic query protocol. v0.1", Gnutella Developer's Forum, May 2003.

[9] L. Garces-Erice, E.W. Biersack, P.A. Felber, K.W. Ross, and G. Urvoy-Keller, "Hierarchical Peer-to-Peer Systems", In Proc. of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par 2003), pp.643-657, 2003.

[10] B. Karp, S. Ratnasamy, S. Rhea, and S. Shenker, "Spurring Adoption of DHTs with OpenHash, a Public DHT service", IPTPS 2004, LNCS Vol. 3279, pp. 195-205, 2004.

[11] S. Kashif, A. Khan and L. Tokarchuk. “Interest-based Self Organization in Group-Structured P2P Networks”, In Proc. of International IEEE CCNC Workshop on Dependable and Sustainable Peer-to-Peer Systems, 2009.

[12] B. Krishnamurthy, J. Wang, and Y. Xie “Early Measurements of a Cluster-based Architecture for P2P systems”. In Proc. of the ACM SIGCOMM Internet Measurement Workshop 2001, pp. 105-109, Nov. 2001.

[13] Q. Lv, S. Ratnasamy, and S. Shenker, "Can Heterogeneity Make Gnutella Scalable", In Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), LNCS Vol. 2429, pp. 94-103, Cambridge, MA, Mar. 2002.

[14] D. Milohicic, V. Kalogeraki, R. Lukose, and K. Nagaraja, J. Pruyne, and B. Richard, "Peer-to-Peer computing", Technical Report HPL-2002-57, HP Lab. Mar. 2002.

[15] S. Noguchi, A Inomata, K. Fujikawa and H. Sunahara. “Efficient Data Management using the Session Log in DHT and its Evaluation”, In Proc. of International IEEE CCNC Workshop on Dependable and Sustainable Peer-to-Peer Systems, 2009.

[16] S. Ratnasamy, S. Shenker, and I. Stoica, "Routing algorithms for DHTs: some open questions". In Proc. of the IEEE International Workshop on Peer-to-Peer Systems (IPTPS), pp. 45-52, Cambridge, MA, USA, Mar. 2002.

[17] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network”, In Proc. of SIGCOMM 2001, pp. 161-172, Aug. 2001.

[18] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: a scalable peer-to-peer lookup protocol for internet applications", IEEE/ACM Trans. Netw., 11(1):17-32, 2003.

[19] A. Singla, C. Ramabhadran, F. Baboescu and A.C. Snoeren, "The Case for Service Provider Deployment of Super-Peers in Peer-to-Peer Networks", Proc. Workshop Economics of Peer-to-Peer Systems, 2003.

[20] A. Singla, C. Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability", Gnutella Developer's Forum, 2002.

[21] Napster(http://www.napster.com/)

[22] Gnutella(http://gnutella.wego.com/)

[23] KaZaa(http://www.kazaa.com)