Query Processing to Efficient Search in Ubiquitous Computing

Byung-Ryong Kim¹ and Ki-Chang Kim²

¹ Department of Computer and Science Engineering, Inha University,
Incheon, Korea

² School of Information and Communication Engineering, Inha University,
Incheon, Korea
{doolyn, kchang}@inha.ac.kr

Abstract. Both ubiquitous computing and mobile ad-hoc networks (MANET) have recently attracted a lot of attention in the research community as well as the industry. Both the domains share certain similarities, primarily the fact that both are instances of self-organizing decentralized systems. Distributed hash table (DHT) provides a very effective and reliable search scheme in Both Networks. However, when the search involves query consist of a set of common words, it suffers heavy network traffic due to the passing around of a large inverted list among nodes. In this paper, we suggest a technique based on the concept of distance between keywords that can remove irrelevant indices from the list. It utilizes the concept of distance between keywords in the query and removes those entries in the inverted list that are going to be dropped sooner or later. We prove this prediction is accurate and effective such that it reduces the size of the inverted list.

Keywords: ubiquitous computing, query processing, distributed hash table.

1 Introduction

Distributed Hash Table(DHT) is used to map a keyword to the peer responsible for holding the corresponding reverse index. Partition-by-Keyword uses DHT to store the indices of files, and uses these tables to find the location of a target document. It hashes the keyword of a document and stores the index of the document (that is its actual location such as URL address) in a node which controls the hash table corresponding to the keyword's hash value. Since a document can contain several keywords, the index of this document can be stored in a number of nodes. To find the location of a document that contains a keyword x , the client peer simply hashes it and send a query to a node which controls the corresponding hash table. However it still poses some performance problems, especially when the query contains many keywords. For query with many keywords, the query should be sent to all nodes that control these keywords. Since the inverted lists (lists mapping the keyword to a list of documents that contain it) from the relevant nodes need to be combined through intersection, the query is passed around with the inverted list being updated at each node. If the first keyword in the query was a common keyword, the starting inverted list

would be huge, and transmitting this huge list to the next node would cause a heavy traffic. Usually the keywords in a particular query consist of a set of common words and another set of specific words that occur only in a limited number of documents. Most of the indices of documents in the starting inverted list, therefore, will most likely be dropped through intersection sooner or later.

It would be beneficial if we can cut down these irrelevant indices from the inverted list before transmitting it to the next node. But how can we know which indices will eventually be dropped? In this paper, we suggest a technique based on the concept of distance between keywords that can remove irrelevant indices from the list. Our technique is explained in detail later in the paper, and the preliminary result shows the technique is very promising. The rest of the paper is organized as follows: Section 2 explains the basic operations of DHT-based P2P system and surveys searching techniques for query with many keywords; Section 3 explains the proposed technique in detail and Section 4 gives a conclusion.

2 Related Works

Numerous researches have been performed to reduce the network traffic in DHT overlay network for keyword search. Multi-level partitioning (MLP) transmits the query only to some limited number of peers to decrease the traffic. In order to select appropriate peers, the technique divides the node space into k logical groups[1]. [2,3] uses Bloom Filter and previous search results to compress the intermediate URI list. Bhattacharjee proposed another technique to improve efficiency: result caching[4]. Keyword Fusion[5] uses Fusion Dictionary, which contains a set of common keywords, to achieve an efficient searching. SCALLOP[6] utilizes a balanced lookup tree to avoid the hot spot problem. In order to reduce query overhead, KSS(Keyword Set Search) by Gnawali partitions the index by a set of keywords[7]. Hybrid-indexing[8] extracts a set of important keywords from the document and utilizes it together with the inverted URI list. mSearch[9] also employs Hybrid-indexing, but it defines a multicast tree for each keyword in order to multicast the query only to some relevant nodes. pSearch[10] reduces the number of visiting nodes for a given query and still achieves high accuracy.

The above techniques have been successful in reducing the network traffic in some aspect, but either the reduction rate is not enough, or they require another type of system resource such as memory space. MLP introduces additional cost for communication between nodes to maintain the grouping information. Using Bloom Filter can cause the problem of false positive in which the hit rate varies considerably depending on the number of hash functions and the size of the bit vector. The dictionary in Keyword Fusion takes time to build and needs to be updated frequently, which costs additional traffic. SCALLOP requires additional storage for lookup table. KSS also causes increasing storage overhead as the number of keyword combination increases. The multicast tree used in mSearch demands additional space overhead, and Hybrid-indexing requires additional space to store major keywords for each document.

3 The Concept of Distance

In this study distance is applied to find out unmatched entry in inverted lists administered by nodes responsible for keyword. Distance means how far the keywords administered in hash range in network are located each other. Node participating in network complies with the following process when distributing document. All the keywords in the document are drawn out and the keywords are hashed using hash function such as SHA-1. Then after arranging the hashed values in descending order, the each difference between two hash values is computed in order.

These computed values are distance. When the number of keywords included in the document is n , the distance of keyword k_i , $d(k_i) = \text{hash}(k_{i+1}) - \text{hash}(k_i)$ and when i is n (keyword with the largest hash value among keywords in the document), $d(k_n) = 0$. Inverted list of keywords included in the document is shown as $\langle \text{document's url}, k_i, d(k_i) \rangle$ added the value of $d(k_i)$. This $d(k_i)$ value makes it possible to tell which keyword with larger hash value than k_i is included in the document including k_i , or if there is any definite chance of inclusion or not. For example to know whether w is included in document x including k_i , if $d(k_i)$ out of every entry of k_i 's inverted list is larger than $(\text{hash}(w) - \text{hash}(k_i))$ it clearly means that w is not included in x , if smaller it means that w may or may not be included, and if it equals to the value, it means w is included for sure.

To apply this notion of distance, the search process shall be different from general one. Let's assume the process of search (x AND y) for document including keyword x and y . First of all keyword x , and y included in query are hashed. If $\text{hash}(x) > \text{hash}(y)$, node y administering inverted list is firstly visited and if not, distance technique can be applied only after visiting node x administering x 's inverted list first. When $\text{hash}(x) > \text{hash}(y)$, the entries, other than with $d(y) > (\text{hash}(x) - \text{hash}(y))$ among y 's inverted list, are forwarded to node responsible for keyword x because it is sure that the excluded entries contain the document URL never including x according to the definition of distance.

Every inverted list of y has been sent to the node responsible for x in the existing techniques but using distance effectively diminishes traffic since only entries surely including x or with high possibility of the inclusion are sent. Tests were conducted to evaluate the performance of the proposed technique, the number of nodes is 1,000 and the number of keywords included in query is randomly set at maximum 5. Test results showed that the traffic was reduced by approx. 40% with 2 keywords in query and the overall traffic was reduced by about 30% on average. The distance forwards the inverted list with any chance of inclusion but the amount of inverted list forwarded this way is not small yet. Hence further study on the distance extension will be needed to solve this problem.

4 Conclusion

DHT is very effective for fast keyword-based-search of a file in ubiquitous computing. However query with many keywords has been known to cause a heavy traffic due to a large inverted list that has to be passed around. We have proposed a technique based on the concept of distance which can cut down the size of the inverted list

considerably and therefore reduce network traffic. According to the test results when the number of keywords included in query was 2 with the application of distance, traffic was reduced by 40%. With 2 to 5 keywords the traffic was reduced by about 30% on average. Consequently with the decrease in the size of inverted list to be sent at intersection at each node, it can say a great contribution to ubiquitous application service.

Acknowledgments. This work was supported by INHA UNIVERSITY Research Grant.

References

1. Shuming Shi, Guangwen Yang, Dingxing Wang, Jin Yu, Shaogang Qu, Ming Chen. Making Peer-to-Peer Keyword Searching Feasible Using Multi-Level Partitioning. In IPTPS, (2004).
2. Patrick Reynolds Amin Vahdat. Efficient Peer-to-Peer Keyword Searching. In Middleware, (2003).
3. B. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7) (1970) 422-426,.
4. B. Bhattacharjee and S. Chawathe and V. Gopalakrishnan and P. Keleher and B. Silaghi, Efficient peer-to-peer searches using result-caching, in The 2nd International Workshop on Peer-to-Peer Systems(IPTPS'03), (2003).
5. Lintao Liu, Kyung Dong Ryu, Kang-Won Lee. Keyword fusion to support efficient keyword-based search in peer-to-peer file sharing, CCGRID 2004, (2004), 269-276..
6. Jerry C. Y. Chou, Tai-Yi Huang, Kuang-Li Huang, Tsung-Yen Chen. SCALLOP: A Scalable and Load-Balanced Peer-to-Peer Lookup Protocol. IEEE Trans. Parallel Distrib. Syst, 17(5) (2006) 419-433.
7. O. Gnawali. A Keyword Set Search System for Peer-to-Peer Networks, Master's thesis, Massachusetts Institute of Technology, (2002).
8. C. Tang and S. Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval, In Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI), (June 2004).
9. Ajay Gulati and Supranamaya Ranjan. Efficient Keyword Search using Multicast Trees in Structured p2p Networks submitted to Middleware (2005).
10. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-organizing Semantic Overlay Networks. In Proceedings of SIGCOMM (2003).