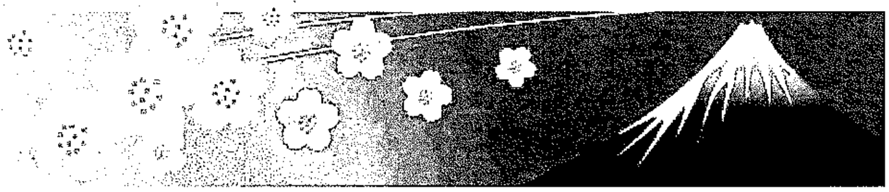
ICDE 2005 TOKYO
International Conference on Data Engineering



Proceedings of the International Special Workshop on Databases For Next Generation Researchers In Memoriam Prof. Yahiko Kambayashi

SWOD 2005

April 4, 2005
Tokyo, Japan

Organized by

The IEEE Computer Society (IEEE CS)

Kyoto University 21st Century Center of Excellence (COE) Program:

Informatics Research Center for Development of Knowledge Society Infrastructure

National Institute of Information and Communications Technology (NICT)

Database Society of Japan (DBSJ)

The Information Processing Society of Japan (IPSJ)

The Institute of Electronics, Information and Communication Engineers (IEICE)

Proceedings
International Special Workshop on Databases for Next Generation Researchers (SWOD2005)

Tokyo, Japan

April 4, 2005

Editors

Kyoji Kawagoe and Tetsuji Satoh

Organized by

The IEEE Computer Society (IEEE CS)
Kyoto University 21st Century Center of Excellence (COE) Program:
Informatics Research Center for Development of Knowledge Society Infrastructure
National Institute of Information and Communications Technology (NICT)
Database Society of Japan (DBSJ)
The Information Processing Society of Japan (IPSJ)
The Institute of Electronics, Information and Communication Engineers (IEICE)

Proceedings of the International Special Workshop on Databases
For Next Generation Researchers(SWOD2005), pp. 84-87(April 2005).

A Query Processing Mechanism for Top-k Query in P2P Networks

Hidekazu MATSUNAMI Tsutomu TERADA
Shojiro NISHIO
Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1–5, Suita-shi, Osaka, 565–0871, Japan
{*matunami.hidekazu, tsutomu, nishio*}*@ist.osaka-u.ac.jp*

Abstract

Recently, there has been an increasing interest in content sharing on peer-to-peer (P2P) networks. Since such a system employs a flooding mechanism for queries and because each peer returns many search results, the system's response to a query creates heavy traffic. Therefore, we propose a new and more efficient query processing method for top-k queries on P2P networks. We focus on the fact that users usually need search results only with a higher score. Our method reduces the reply traffic by controlling the number of query replies. Moreover, we evaluate the proposed method by simulation studies.

1. Introduction

Recently, the amount of web content on the World Wide Web (WWW) has been increasing rapidly. Users can find required content efficiently within a mass of web content by using a web-search engine such as Google[3]. However, we cannot get search results or find web content when servers or networks are in poor condition.

On the other hand, there has been increasing interest in the research of content sharing on peer-to-peer (P2P) networks. We can share resources or services directly through connections among computers with a P2P-based contents sharing system, which does not need a centralized server.

Generally, when we request certain web content, we use a full-text search system with specified keywords. Usually, we get many search results with a full-text web search system. On server-client-based systems such as Google, the system can easily choose the best-k answers for the query because it manages all of the contents. On the other hand, on P2P-based systems, each peer sends query-replies, and this may cause a large amount of traffic at query-issuing peers.

In this paper, we propose efficient methods for realizing top-k queries on a P2P-based content retrieval system. In these methods, we reduced the number of query-replies to a request according to the number of hops from the query-issuing peer, and in this way we can reduce the traffic of query-replies. Also, we evaluate the proposed method by simulation studies.

The rest of the paper is organized as follows. In Section 2, we introduce related work to this research. Section 3 presents three methods for realizing the top-k queries. Section 4 describes the simulation results. Section 5 gives concluding remarks.

2. Related Work

Pure P2P-based systems, which do not use a centralized server, are divided into systems with structured search topologies and those with unstructured search topologies.

CAN[6] and Chord[7] are P2P-based systems with structured retrieval topologies. These systems strictly provide the distribution of a contents key in hash-space by using distributed hash tables (DHT). The file name or keywords of a content item are used to create the contents key.

Closer to our work is the pSearch[8], a decentralized non-flooding P2P information retrieval system based on CAN. pSearch distributes a document's indices through the P2P network based on the document's semantics, which are generated by Latent Semantic Indexing (LSI). However, in these systems, P2P networks have to be strictly organized.

Gnutella[2] and Freenet[4] are systems using unstructured search topologies. Such systems have no special restriction on network topologies and contents-distribution. Thus, there is the advantage that peers do not need to exchange information before searching. However, to get a more accurate search result, a peer has to send queries to more peers. This brings about a larger network load compared with structured retrieval topologies.

Kalnis[5] proposes a system that realizes a top-k query on a Gnutella-type P2P network. In this method, when executing a top-k query, each peer receiving the query returns a number of query-replies about top-k contents. Our proposed methods enhance the efficiency of this method.

Balke[1] proposes another method for processing top-k queries. This method relies on a super-peer backbone organized in the HyperCuP topology. In this method, each peer receiving one query returns only one query-reply. When the query-issuing peer needs more query-replies, the peer sends the query again. This method can minimize traffic, especially with a small k. However, when k is large, query latency becomes extended.

3. Proposed Methods

In this section, we show the three Top-k query-processing methods proposed in this research.

3.1 Simple top-k method

The simple top-k method is one in which each peer simply replies to a query. The difference between this method and the one proposed by Kalnis is that each peer transfers query-replies only within the top-k query-results from the received query-replies and contents discovered within itself.

First, a query-issuing peer, which requests certain content, makes a search-query that consists of a unique query-id, the search terms, the length of time-to-live for the query (TTL), and k, the number of query-replies that the peer needs. Then, the query-issuing peer sends the search-query to all neighboring peers, that is, peers directly connected to the query-issuing peer. Next, the query-issuing peer retrieves the top-k contents within itself and creates a top-k search-result list.

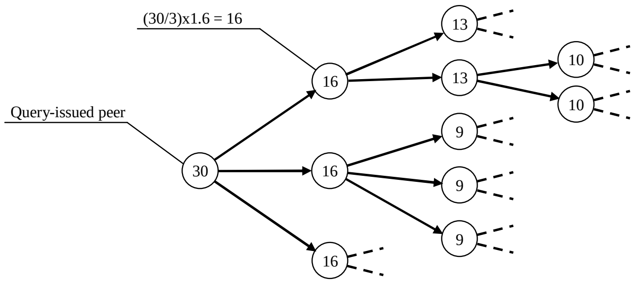
Each peer receives the query and forwards the query after decreasing the TTL. Next, it retrieves contents and returns the query-replies with the top-k contents to the peer from whom it received the query.

Each peer that receives query-replies forwards the query-replies only if they are included in the top-k search-results list.

With this method, the query-issuing peer can certainly get the top-k query results from query-reach peers, which are the peers that received the search-query. However, low-ranked content among the top-k on each peer has a small possibility of being included in the top-k contents on the query-issuing peer, which means that too many unnecessary query-replies could overflow the network.

3.2 Reduce-k query method

The reduce-k query method reduces the number of query-replies on each peer to reduce the network load. This method is almost idenitical to the simple top-k method without the number of query-replies sent by each peer.



($k = 30,\ \ r_m = 1.6$, Number in circles is k_i.)

Figure 1. Number of query-replies for each peer.

We assign k_i as the number of query-replies sent by a peer leaving i hops from the query-issued peer. r_m is the margin of query-replies, and n_{p_i} is the number of peers to which a peer forwards the query. We then define the temporary number of k_i, i. e. k'_i, as below:

$$k'_i = \left\lfloor \frac{k_{i-1} \times r_m}{n_{p_i}} + 0.5 \right\rfloor \tag{1}$$

However, we cannot get a query-reply from a peer where k_i is zero. Moreover, if k_i is more than k_{i-1}, it is useless because a parent peer sends query-replies no more than k_{i-1}. Consequently, we regularize k_i as below:

$$k_i = \begin{cases} k_{i-1} & (k_{i-1} \le k'_i) \\ k'_i & (2 \le k'_i < k_{i-1}) \\ 2 & (k'_i < 2) \end{cases} \tag{2}$$

We show an example of determining k_i in Figure 1. k_i may be different when the number of hops from the query-issuing peer are the same.

3.3 Delayed reduce-k query method

The delayed reduce-k query method is one where each peer makes a delay in sending or forwarding query-replies. We define the time of delay t_{l_T} as the time from the moment the query-reply is finally received. The number of t_{l_T} is defined as below:

$$t_{l_T} = t_0 + t_1 T \tag{3}$$

In this expression, t_0 and t_1 are the parameters determined in a search-query. T is the number of TTL on the search-queries transferred to neighboring peers.

In this method, each peer waits to forward or send query-replies for t_{l_T} seconds. If content is not included in the top-k search-result list before it is time to send query-replies, there is no need to send a query-reply with such content.

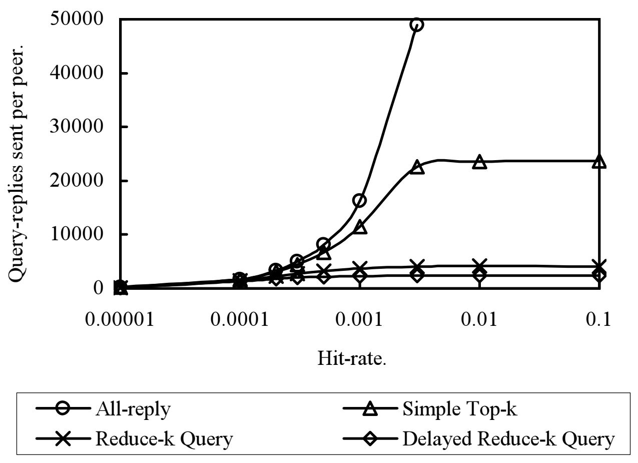


Figure 2. Hit-rate vs. query-replies.

We can also define the immediate query-reply rate as r_i. Each peer sends query-replies on the top-$(k_i r_i)$ of the top-k search results list without any delay.

4. Evaluation

In this section, we show the results of our simulation.

4.1 Environmental assumption

In this paper, we assume a top-30 full-text web search on P2P networks.

The number of peers in the simulation environment is set to 10,000. Each peer connects in a ring configuration: No.0 to No.1, No.1 to No.2, and so on. Each peer selects one peer randomly and connects to it. In this paper, we assume that the network topology is fixed.

The number of content units is 1 million, and each peer has 10,000 of them. We distribute content along a Zipf-like distribution.

In this simulation, each peer issues a query lasting 1,000 seconds. Simulation time is defined as 1,000 seconds.

The communication speed between peers is 512 kbps, and the latency of messages is 0.001 + 0.01 $\times$ ExpRnd seconds. (ExpRnd is an exponential random number whose average is 1.) The length of the query message is 140 bytes, and the length of the query-reply message is 640 bytes.

4.2 Simulation results

We ran the simulation in the environment shown above. We show the average number of query-replies sent by each peer when we change a hit rate which returns all of the

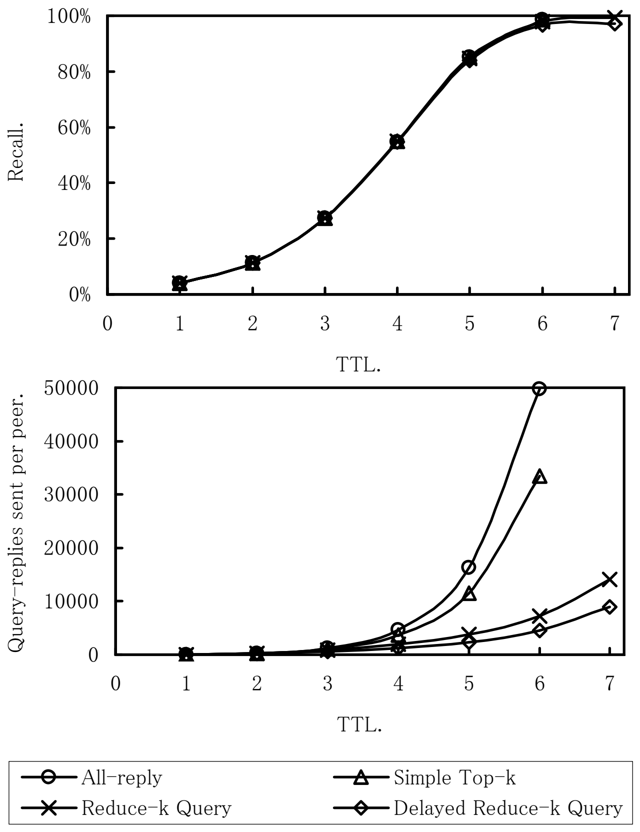


Figure 3. TTL vs. recall and query-replies.

content satisfying the search conditions, for the simple top-k method, the reduce-k method, and the delayed reduce-k method. We set k to 30 in the simple top-k method. In the reduce-k query method and the delayed reduce-k method, we set k to 100 and r_m at 1.5 because we could get better results than when k is 30. In the delayed reduce-k method, we set the query-latency time to $0.01 + 0.002T$, while T is the TTL on a peer.

Figure 2 shows the average number of query-replies for each peer when the hit rate is changed, which is the probability of content hitting a query, between 0.00001 and 0.1, while the TTL is fixed to 5. The number of query-replies in the three proposed methods reach the ceiling when the hit-rate reaches a certain value. We reduce 54% of the network load using the simple top-k method compared with the all-reply method when the hit-rate is 0.003. Moreover, on the ceilings, we reduce the network load using the reduce-k query method and the delayed reduce-k query method by 83% and 89%, respectively, compared to the results achieved via the simple top-k method.

Next, Figure 3 shows the recall and the average number of query-replies sent by each peer by changing the TTL to between 1 and 7 while the hit-rate is fixed to 0.001. Re-

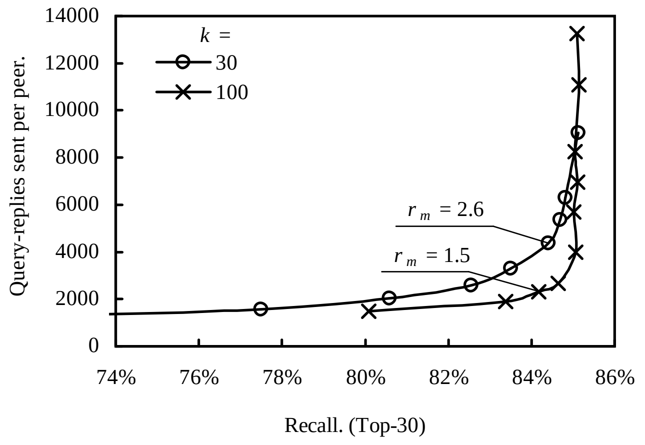


Figure 4. Recall vs. query-replies.

call gains by the TTL show an increase, and reach a sufficient value when the TTL is 6. Furthermore, there is little difference in recall between the all-reply method and the three proposed methods. This shows that the three proposed methods have sufficient recall when we set the parameters, k and r_m, properly.

As expected, the number of query-replies sent by each peer grows exponentially when the TTL increases. However, despite the number of query-replies increasing by about three times in the all reply method and the simple top-k method, the number of query-replies increases only twice on the reduce-k query method and the delayed reduce-k query method. This means it is easy to increase the TTL in the reduce-k query method and the delayed reduce-k query method.

Finally, we show why we set k to 100 despite the assumed environment having a top-30 full-text web search. Figure 4 shows the relationship between the number of query-replies sent by a peer and the recall on a top-30 search Figure 4 shows that we can reduce the number of query-replies sent by a peer by setting k to a larger number while still achieving the same recall. For example, supposing we need 84% recall, we have to set r_m to 2.6 when k is 30, while when k is 100, 1.5 is a sufficient value for r_m. In this situation, we can reduce 42% of the query-replies by setting k to 100.

5. Conclusion and Future Work

In this paper, we proposed three methods to process top-k queries efficiently on P2P networks. We evaluated the performance of the proposed methods by simulation. Using the proposed methods, the system can reduce the large number of query replies sent by a peer under the condition of a constant number.

However, in the reduce-k query method and the delayed reduce-k query method, we cannot reliably get top-k results. In some cases we can get only 60% or 70% of query-replies compared to the simple top-k method.

In the future, we will evaluate our method in an environment where the network topology changes dynamically. Moreover, we will propose another method which has the ability to return all query-replies with a smaller network load than that of the simple top-k method.

Acknowledgement

This research was supported in part by a Grant-in-Aid for Scientific Research (B) (15300033) from the Ministry of Education, Culture, Sports, Science and Technology of Japan and as part of the 21st Century Center of Excellence Program of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks," Technical Report of Hannover University, July 2004.

[2] "Gnutella," http://www.gnutella.com/.

[3] "Google," http://www.google.com/.

[4] I. larke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Proc. of ICSI Workshop on Design Issues* in Anonymity and Unobservability, pp.46–66, July 2000.

[5] P. Kalnis, W. S. Ng, B. C. Ooi, and K. -L. Tan, "Answering Similarity Queries in Peer-to-Peer Networks," *Proc. of International World Wide Web Conference*, pp. 482–483, May 2004.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. of SIGCOMM'01*, pp. 161–172, Aug. 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Application," *Proc. of SIGCOMM'01*, pp. 149–160, Aug. 2001.

[8] C. Tang, Z. Xu, and M. Mahalingam, "pSearch: Information Retrieval in Structured Overlays", ACM SIGCOMM Computer Communications Review, vol. 33, no. 1, pp. 89–94, Jan. 2003.