

# Exploiting Semantics in Unstructured Peer-to-Peer Networks

Kiyohide NAKAUCHI<sup>†a)</sup>, Yuichi ISHIKAWA<sup>††\*</sup>, Hiroyuki MORIKAWA<sup>†††</sup>,  
and Tomonori AOYAMA<sup>††††</sup>, Members

**SUMMARY** Decentralized and unstructured peer-to-peer (P2P) networks such as Gnutella are attractive for large-scale information retrieval and search systems because of their scalability, fault-tolerance, and self-organizing nature. Because of this decentralized architecture, however, traditional P2P keyword search systems are difficult to globally share useful semantic knowledge among nodes. As a result, traditional P2P keyword search systems cannot support semantic search (support only naive text-match search). In this paper, we describe a design of the semantic P2P keyword search system. We exploit the semantics of correlation among keywords rather than synonym. The key mechanism is *query expansion*, where a received query is expanded based on *keyword relationships*. Keyword relationships are improved through search and retrieval processes and each relationship is shared among nodes holding similar data items. This semantic P2P search system has two main advantages. First, expanding search results through query expansion increases the possibility of locating desired data items which would not be found by traditional P2P search systems due to the keywords' textual mismatch. Second, keyword relationships originally introduced for query expansion, can be used for result ranking. Our main challenges are 1) managing keyword relationships in a fully decentralized manner and 2) maintaining the quality of search results, while suppressing result implosion. We also describe the prototype implementation and evaluation of the semantic P2P search system.

**key words:** peer-to-peer network, distributed keyword search, semantic search

## 1. Introduction

As the number of Web content items grows into the billions, a larger and larger portions of the content is not accessible through centralized search engines [1], [2]. Recent estimates (March 2000) [3] indicate that deep Web is nearly 500 times the size of the publicly indexable Web. The deep Web continues to grow at a remarkable rate as more existing databases become available online for which Web-accessible search facilities are provided. This means there is a need for search infrastructure scaling at a comparable rate.

Peer-to-peer (P2P) systems are now one of the most prevalent Internet distributed applications because of their scalability, fault-tolerance, and self-organizing nature. This

trend was triggered in 1999 by Napster [4], a centralized architecture, where a central directory server offered an index to locate data items. However, the legal issues that Napster faced because of its centralized architecture shifted the interests of the research community and Internet users to a decentralized architecture, where the index, query processing, and content transfer are fully distributed among nodes.

The problems of decentralized architectures that developers primarily focus on are related to the scalability and partial-match lookup (keyword search) capability. One class of decentralized architectures is an unstructured P2P system such as Gnutella [5], where the overlay topology is formed in accordance with some loose rules [6]. Data items are indexed locally and a query can be resolved only by the nodes that hold the data items. Due to this loose structure, queries are typically flooded. Thus, unstructured P2P systems are simple enough to support partial-match lookups without any other complicated mechanisms and very robust against dynamic node failure and removal. Though P2P systems based on query flooding are generally considered unscalable based on traffic analysis [7], efforts have recently been made to improve scalability. For example, FastTrack [8] organizes subscribing nodes into a loosely hierarchical structure where some nodes are selected as supernodes and cache the index. Multiple random walks [9], associative overlays [10], interest-based shortcut [11], and YAPPERS [12] reduce the number of forwarded queries by limiting searches to a fraction of participants rather than blind flooding. Gia [13] enhances scalability by the synergy of random walks, topology adaptation, and token-based flow control.

The other class of decentralized architectures is a structured P2P system commonly referred to as Distributed Hash Tables (DHTs) [14]–[17], where the overlay topology is tightly controlled. The nodes that are required to store or index data items are precisely determined based on some hashing algorithms. This tightly controlled structure enables the forwarding of queries deterministically and achieves very effective content location. While structured P2P systems are highly scalable for exact-match lookups—that is, for locating data items with a unique identifier—inherently they cannot efficiently provide a partial-match lookup capability. Some work [18]–[20] have been aimed at providing a partial-match lookup capability on DHTs through a sophisticated method of generating keys (globally unique identifiers or GUIDs) corresponding to data from the attached multiple keywords. In addition, it is widely recognized that their

Manuscript received November 19, 2003.

Manuscript revised January 23, 2004.

<sup>†</sup>The author is with Communications Research Laboratory, Koganei-shi, 184–8795 Japan.

<sup>††</sup>The author is with School of Engineering, The University of Tokyo, Tokyo, 113–8656 Japan.

<sup>†††</sup>The author is with School of Frontier Sciences, The University of Tokyo, Tokyo, 113–8656 Japan.

<sup>††††</sup>The author is with School of Information Science and Technology, The University of Tokyo, Tokyo, 113–8656 Japan.

\*Presently, with NEC Corporation.

a) E-mail: nakauchi@crl.go.jp

tight control means that DHTs incur a much larger overhead than unstructured P2P systems when the overlay network is re-organized because nodes fail or leave the network.

As described above, current P2P systems are being actively improved through recent research to achieve both scalability and a partial-match lookup capability. However, the search algorithms of current P2P systems are still not sufficiently efficient because the current P2P keyword search systems are difficult to globally share useful semantic knowledge, such as regarding the popularity of data items and the relationships between keywords and data items, among nodes. That is, while current P2P systems support naive text-match search, they cannot support *semantic search*. As a result, the systems can only find data items which are given a keyword (or metadata) exactly indicated in a query.

The most familiar mechanism enabling semantic search is *query expansion* [21],[22], which has been investigated as an IR (Information Retrieval) technique for several decades. Query expansion means adding relevant terms to the original query. The purpose of query expansion is to cope with any mismatch between the term used by a searcher and that expected by writers of the documents. This mismatch may be due to synonymy, where different terms have the same meaning, or granularity, where terms are used at different levels of detail. We believe that in P2P search systems, query expansion can significantly improve the possibility of locating desired data items because most of the contents which current popular P2P search systems deal with are multimedia contents, which generally have a fairly small number of keywords.

Our goal is to build an efficient decentralized P2P search system that supports semantic search through query expansion, while retaining the desirable properties of current unstructured P2P systems such as simplicity and robustness. Our system enables efficient location of not only well-defined items (those given keywords that are easy for searchers to think of and that can be located by traditional text-match P2P search systems), but also of poorly-defined items which have keywords that are generally or unexpectedly difficult for searchers to think of.

We faced two main challenges when designing our semantic P2P keyword search system. The first is to construct and manage the databases required for query expansion in a fully decentralized manner. In traditional IR systems, a query is generally expanded based on some kind of database, such as a thesaurus, which contains relationships between terms extracted from the statistics computed in advance using samples representative of a sufficient amount of documents. In P2P keyword systems, however, because of the decentralized nature, it is undesirable to have a centralized node calculate and maintain the statistics to obtain keyword relationships. Even if a complete thesaurus-like database exists, we believe that such a database (which may be large enough for a node to store) should be divided and each portion kept among distributed participants to retain the general advantages of P2P systems such as robustness

and load balancing. We also believe that keyword relationships, though they may change at relatively slow rate, should be computed in a distributed manner for the same reason.

The second challenge is to maintain the quality of search results while suppressing “result implosion” in the worst case, which is the explosive increase of returned search results due to query expansion. To cope with this problem, we introduce a results ranking mechanism where data items with more keywords relevant to a query are ranked higher. Fortunately, we can exploit keyword relationships for this purpose by introducing the concept of the “strength” of each keyword relationship.

In this paper, we describe the basic design of a semantic P2P keyword search system using keyword relationships. We also describe the distributed mechanisms of updating the databases used for query expansion. These databases, which we call Keyword Relation DataBases or KRDBs, are managed by each node and store the keyword relationships relevant only for the shared local data items the node holds. As sophisticated KRDBs can significantly improve search results, we propose two KRDB update mechanisms: evaluation feedback and KRDB synchronization. In addition, we show a simple ranking mechanism which uses the keyword relationships originally introduced for query expansion.

The notable advantage of our semantic P2P search system is that the search results are as accurate as those of traditional systems while including even poorly-defined desired data items. In other words, all well-defined data items can be located and are ranked higher, and poorly-defined data items can be located but are ranked in accordance with the strength of the relationship between keywords in a query and the data items. Thus, locating poorly-defined data items has little effect on the ability to locate well-defined items. Several features that differentiate our semantic P2P search system from others are as follows:

- Our system is controlled in a fully distributed fashion, without any single point of failure and any centralized management and placement of KRDBs.
- Our system supports semantic search in addition to naive text-match search, and improves the possibility of locating desired data items. We exploit the semantics of “correlation” among keywords rather than “synonymy.”
- Our system ranks search results by exploiting keyword relationships.

Before we proceed, we would like to emphasize the following points. First, this paper is not concerned with how to improve the quality of search results to the comparable degree of state-of-the-art IR algorithms developed for centralized search engines such as Google. Recall our concern is improvement of the possibility of locating poorly-defined data items. Second, we use the word “semantics” to define correlation rather than synonymy between any two keywords. Therefore two correlative keywords do not necessarily have synonymy. Our semantic P2P search system does not cope with misspelling. Third, the classification of our

target content is multimedia data such as music files rather than textual data. In addition, we do not assume a specific naming standard and metadata framework for data items. We also assume metadata are typically attached manually by experts or only limited types of metadata are attached automatically. On these assumptions, multimedia data items are generally attached fewer keywords than textual ones. For such data items, it is well known that traditional IR algorithms do not work so accurately. This discussion motivates us to exploit the alternative semantics (correlation of keywords). Finally, while this paper shows a whole P2P search system, the concepts added to the original P2P search system (Gnutella) such as KRDBs, their synchronization, and results ranking could be applied to other recently developed P2P search systems with scalability or search efficiency. Therefore, we do not discuss on query routing or the overlay topology in this paper.

The reminder of this paper is organized as follows. In Sect. 2, we give an overview and describe the basic mechanisms of a P2P keyword search with query expansion based on keyword relationships. Section 3 describes the distributed update mechanisms of KRDBs to enhance search performance. We describe the implementation and evaluation of our P2P search system in Sect. 4 and in Sect. 5, respectively. We discuss various issues regarding the proposed P2P keyword search system in Sect. 6. We show related work in Sect. 7, and conclude in Sect. 8.

## 2. Semantic P2P Search

The fundamental idea in our P2P keyword search mechanism is query expansion using a KRDB that is managed and updated in a fully distributed manner. In this section, we explain how to create a KRDB and then describe the basic search mechanism. We also describe the ranking algorithm.

In our P2P keyword search system, as in a traditional unstructured P2P system, the overlay topology is organized in accordance with some loose rules [6]. A query includes several keywords and is flooded to be resolved. This unstructured architecture enables the system to retain desirable properties such as simplicity and robustness against node's failure or removal.

Before we describe the details, we simply define a "keyword." In general, meta-data are attached to data items in the form of an attribute/value pair. In this paper, we refer to the value as a keyword.

### 2.1 KRDB

A KRDB is a thesaurus which keeps some information about the keywords relevant only to the data items stored locally in a node. That means each node may have a different and minimum KRDB. This distributed KRDB management can clearly retain the desirable properties of P2P systems.

The most important information on keywords in a KRDB is the *keyword relationship* (KR) of each pair of keywords and its strength. In this paper, we refer to the key-

word relationship itself from keyword  $k_i$  to keyword  $k_j$  as  $KR(k_i, k_j)$  ( $1 \leq i, j \leq n$ , where  $n$  denotes the maximum number of keywords in a KRDB). In other words,  $KR(k_i, k_j)$  is defined as follows; when keyword  $k_i$  is given, keyword  $k_j$  is referred to as a relevant term to keyword  $k_i$ . Note that  $KR(k_i, k_j)$  and  $KR(k_j, k_i)$  should be distinguished from each other. The other variables in a KRDB are shown in Sect. 3.

Before we explain the algorithm of extracting KRs, we describe our insight. Our insight is such that keywords given to a data item are relevant to one another. In the current P2P search systems, we assume data items are mainly multimedia contents such as audio and video, which have generally much fewer keywords than documents because they have much less textual contents. Therefore, we consider these keywords represent the characteristics of the data items more precisely and keyword relationships would be helpful for finding a limited number of other meaningful keywords. In addition, the extracted KRs are just initial state, and gradually improved through KRDB update mechanisms shown in Sect. 3.

Figure 1 shows how to initially create KRs between keywords. There are two processes to create KRs. First, when a node joins the P2P network, the node firstly extracts all the keywords for each local data item. For example, the node takes out four keywords A, B, C, D from data item 1. We consider these keywords have relationships between each other. Following the definition of  $KR(k_i, k_j)$ , twelve KRs are created from data item 1. In the same way, twelve and six KRs are created from data item 2 and 3 respectively. Each  $KR(k_i, k_j)$  keeps  $KRStr(k_i, k_j)$ , which denotes the normalized variable of strength of  $KR(k_i, k_j)$  ( $0 \leq KRStr(k_i, k_j) \leq 1$ ). Larger  $KRStr(k_i, k_j)$  means  $KR(k_i, k_j)$  is stronger.  $KRStr$  is updated to reflect more accurate KR based on both evaluation feedback and KRDB synchronization described in Sect. 3, and is also used for results ranking. The initial value of  $KRStr$  is  $KRStrInit$  (0.5 in our system).

To further sophisticate a KRDB, additional KRs are created if two KRs share the same keyword. For example, as shown in Fig. 1, if the value of  $KRStr(F, E) \times KRStr(E, I)$  is larger than the pre-defined threshold  $KRStrThresh$ ,  $KR(F, I)$

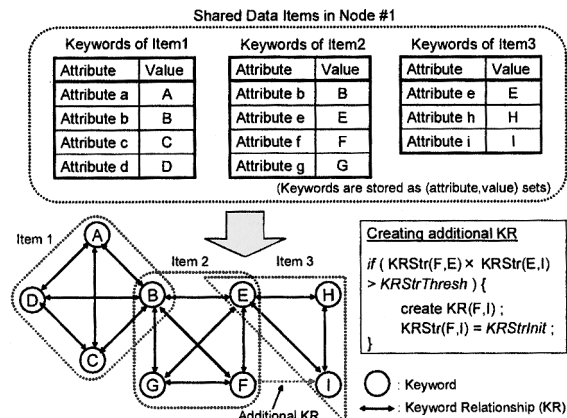


Fig. 1 Creating keyword relationships from local data items.

is newly created. However, less useful KRs are removed from KRDBs to prevent the waste of storages and computation power (see Sect. 3.2).

These two KR creation processes are called when local data items are modified or newly added, or when a local KRDB is improved through KRDB updates mentioned in Sect. 3.2.

### 2.2 Basic Search Mechanism

The key mechanism of our P2P keyword search system, which differentiates our system from traditional unstructured P2P systems, is query expansion at a node. With regard to where a query is expanded, there are two approaches: sender-side expansion and receiver-side expansion. The former expands a query only at a sender (searcher) when it generates the query. This approach, however, works well only when a searcher uses the keywords kept in its local KRDB, because no node has a complete KRDB keeping all existing keywords. Otherwise, the query is not expanded anywhere, which leads to the same search results as those of traditional non-semantic P2P search. On the other hand, the latter expands an original query at the nodes that receive the query. In this approach, a query is expanded at all the adequate nodes that keep relevant keywords to the query. Thus query expansion is not limited to the keywords in the searcher’s local KRDB. This discussion leads us to adopt receiver-side expansion.

Figure 2 shows the basic search mechanism using query expansion. When a node join the P2P network, the node first constructs a KRDB in the way described in Sect. 2.1. The search process is as follows.

1. A searcher issues a query which indicates several keywords. This query is flooded (forwarded in a P2P manner like Gnutella) with a certain TTL (Time To Live). Note that the forwarded query is identical with the received query (query expansion affects only local search), because consecutive query expansion at different nodes leads to query explosion with less relevant keywords, so that the possibility of finding less desired data items increases. (Process ① in Fig. 2)

2. A node which receives a query performs query expansion using its local KRDB. Specifically, the original query is expanded to include several additional keywords to which there are KRs from keywords in the original query. For example, in Fig.2, Node #1 keeps KR(red, apple) in its local KRDB (with KRStr(red, apple) = 0.6) so that a keyword “red” is expanded to two keywords, “red” and “apple.” In the same way, keyword “fruits” is expanded to two keywords, “fruits” and “apple.” Then all the expanded keywords for each original keyword are merged. After all, the expanded query includes three keywords, “red,” “fruits,” and “apple,” with which Node #1 searches for local data items. (Process ②)
3. If a data item exists that has one or more keywords of the expanded query, Node #1 replies to the searcher with search results (a list of satisfied data items) using a QueryHit message. (Process ③)
4. The searcher gathers search results and ranks all the located data items. The ranking algorithm is described in Sect. 2.3. The searcher then selects one or more desirable data items, and the search itself is completed. (Process ④)
5. At the same time, the searcher feedbacks the evaluation to all the nodes which returned search results obtained using KR(red, apple) or KR(fruits, apple) (Node #1 and #2 in Fig. 2) for the purpose of updating KRDBs in those nodes. Evaluation results indicate which KRs were used to locate the selected data items. The details of evaluation feedback are described in 3.1. (Process ⑤)

### 2.3 Results Ranking

The search results are ranked based on KR strength between a keyword in an original query and that given to the located data item. The basic idea of results ranking is as follows; When an original query includes keyword  $k$ , the rank of the data item with keyword  $l$  gets higher as the value of  $KRStr(k, l)$  in the receiver’s KRDB is larger. If several keywords are included in a query, or are given to data items, the above ranking algorithm is changed as follows; when an original query includes keywords  $k_i$  ( $i = 1, 2, \dots$ ), the rank of the located data item gets higher as simply  $\sum_{i, l'} KRStr(k_i, l')$  is larger, where  $l'$  ( $i' = 1, 2, \dots$ ) denote keywords given to the data item. Note that other sophisticated ranking algorithms using  $KRStr$  could be applied. We believe these ranking algorithms would also benefit traditional P2P search systems, but only for the purpose of results ranking by introducing KRDB-like databases otherwise unnoticeable for users into the search systems. Such an application, though, is beyond the scope of this paper.

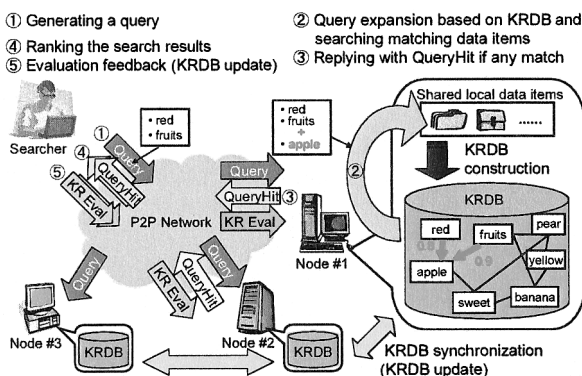


Fig. 2 Basic P2P search mechanism.

### 3. Distributed KRDB Updates

In our P2P keyword search system, the accuracy of KRDBs significantly affects the search performance. Therefore KRDBs are required to be updated and kept as accurate as possible. To retain the desirable properties of unstructured P2P systems such as simplicity and robustness, we should design KRDBs updates to be managed in a fully distributed manner. In this paper, we show two distributed KRDB update mechanisms; evaluation feedback and KRDB synchronization. Evaluation feedback is aimed at improving KRDBs through a search process. In this mechanism, the subjective evaluations of searchers are directly reflected in the KRDBs, and potential statistical effects can be expected. KRDB synchronization is aimed at improving KRDBs that include inaccurate information due to special circumstances: for example, when a node has just joined the P2P network, or a node has just exposed many new data items. These two update mechanisms are complementary to each other and are essential for keeping KRDBs accurate.

#### 3.1 Evaluation Feedback

The evaluation feedback updates  $KRStr(k_i, k_j)$  in the nodes which replied to the searcher with resulted data items found using  $KR(k_i, k_j)$ . The basic idea of evaluation feedback is as follows. When a searcher initiates a query with keyword  $k_i$  and then selects a data item with keyword  $k_j$  from the resulted item list (the item was located by query expansion using  $KR(k_i, k_j)$ ),  $KR(k_i, k_j)$  is regarded as helpful and  $KRStr(k_i, k_j)$  is increased. Otherwise,  $KRStr(k_i, k_j)$  is decreased.

In Table 1, we show the variables kept in a KRDB.  $UsedCnt(k_i, k_j)$  is incremented when both of the following conditions are satisfied; 1) the original query including keyword  $k_i$  is expanded using  $KR(k_i, k_j)$  and 2) a data item with keyword  $k_j$  is located. However,  $HelpfulCnt(k_i, k_j)$  is incremented only when a third condition is also satisfied; 3) a data item with keyword  $k_j$  is selected by a searcher. We define  $KRStr(k_i, k_j)$  using these two variables;

$$KRStr(k_i, k_j) = \frac{HelpfulCnt(k_i, k_j)}{UsedCnt(k_i, k_j)}$$

This means that  $KRStr(k_i, k_j)$  increases as more searchers regards  $KR(k_i, k_j)$  as useful one ( $HelpfulCnt(k_i, k_j)$  increases). Currently, the initial values of each variable are set as follows:  $UsedCntInit = 2$ ,  $HelpfulCntInit = 1$ ,  $KRStrInit = 1/2$

**Table 1** Variables in a KRDB.

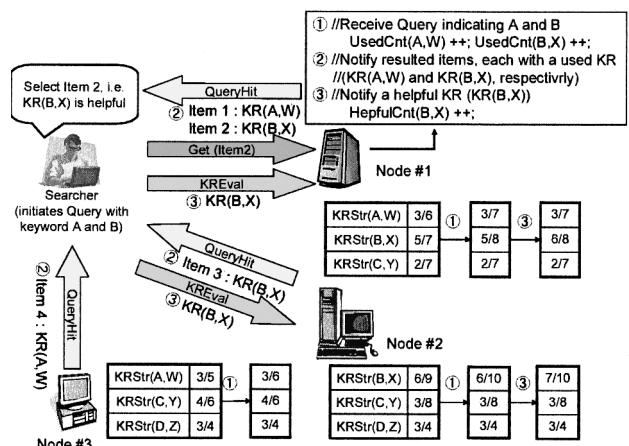
$PKFlag(k_i)$	If $k_i$ is PK, $PKFlag(k_i) = 1$ . Else, $PKFlag(k_i) = 0$ .
$KRStr(k_i, k_j)$	Strength of $KR(k_i, k_j)$ . ( $0 \leq KRStr(k_i, k_j) \leq 1$ )
$UsedCnt(k_i, k_j)$	The count that $KR(k_i, k_j)$ is used for search. Initial value $UsedCntInit = 2$ .
$HelpfulCnt(k_i, k_j)$	The count that $KR(k_i, k_j)$ is evaluated to be helpful. Initial value $HelpfulCntInit = 1$ .

= 0.5.

Figure 3 shows the process of evaluation feedback. In this figure, a searcher initiates a query that includes keywords A and B.

1. Each node receives a query with keyword  $k_i$  from a searcher. At this time, a node which keeps  $KR(k_i, *)$  (\* denotes an arbitrary keyword) increments  $UsedCnt(k_i, *)$ . In Fig. 3,  $UsedCnt(A, W)$  and  $UsedCnt(B, X)$  are incremented at Nodes #1 and #3, and at Nodes #1 and #2 respectively. More specifically, for example, Node #1 initially has  $KRStr(A, W) = 3/6 (= 0.5)$ , which means  $UsedCnt(A, W) = 6$  and  $HelpfulCnt(A, W) = 3$ . At Node #1,  $UsedCnt(A, W)$  and  $UsedCnt(B, X)$  are then incremented from 6 to 7 and from 7 to 8, respectively.
2. Each node notifies the searcher of search results with  $KR(k_i, *)$  used for locating the data item. In Fig. 3, for example, Node #1 notifies the searcher of Item 1 with  $KR(A, W)$  and Item 2 with  $KR(B, X)$ .
3. When a data item without keyword  $k_i$  (with keyword  $k_j$ ) is selected by the searcher among search results, the searcher notifies all the responsive nodes by unicast who keep the same KR ( $KR(k_i, k_j)$ ) used to locate the data item. At this time, the node that receive this evaluation feedback increments  $HelpfulCnt(k_i, k_j)$  of the KRs specified by the feedback.

In Fig. 3, the searcher selects data item 2 among search results. Data item 2 is found using  $KR(B, X)$  at Node #1, so the searcher refers to  $KR(B, X)$  as a helpful relationship. The searcher then sends evaluation feedback (using a  $KREval$  message) to Nodes #1 and #2, who return the data item found using  $KR(B, X)$ . When Nodes #1 and #2 receive evaluation feedback, they increment  $HelpfulCnt(B, X)$  from 5 to 6, and from 6 to 7 respectively. Note that  $HelpfulCnt(A, W)$  retains the same value because  $KR(A, W)$  is useless for the searcher. Thus, the strength of the more helpful relationship ( $KRStr(B, X)$ ) increases, while that of the less helpful relationship ( $KRStr(A, W)$ ) decreases.  $KRStr(C, Y)$



**Fig. 3** KRDB update through evaluation feedback.

and  $KRStr(D, Z)$  remain unchanged because keywords C and D are not included in the query.

Through this evaluation feedback process of all the searchers, each  $KRStr$  in  $KRDBs$  is gradually and statistically refined.

### 3.2 KRDB Synchronization

The  $KRDB$  update mechanism by evaluation feedback is an efficient way to improve the accuracy of  $KRStr$ . However, evaluation feedback has two drawbacks. First, evaluation feedback can only evaluate the existing  $KRs$ , which are extracted only from local data items. Second, while evaluation feedback can basically improve the accuracy of  $KRDBs$ , it would take a long time to make the value of  $KRStr$  statistically meaningful. For this reason, the nodes with a short lifetime or nodes that have just begun to expose new data items cannot soon provide accurate  $KRDBs$ .

To overcome these drawbacks, we propose another  $KRDB$  update mechanism,  $KRDB$  synchronization, where familiar  $KRs$  and a statistically more accurate value of  $KRStr$  are shared among the nodes. The basic idea of  $KRDB$  synchronization is as follows; 1)  $KRs$  relevant to a node are added to the node’s  $KRDB$ , and 2) when the same  $KRs$  are shared at some nodes, the value of  $KRStr$  at each node is updated to the most accurate value.

Note that  $KRDB$  synchronization does not guarantee consistency among  $KRDBs$  in remote nodes but only improves them if two nodes fortunately share the same  $KRs$ . This loose rule does not degrade the robustness of unstructured P2P networks.

#### 3.2.1 Which Nodes are Selected for $KRDB$ Synchronization?

The simplest approach is to synchronize  $KRDBs$  with all participating nodes in the P2P system. However, it is clear that this approach will not scale at all.  $KRDBs$  must therefore be synchronized with only a limited number of nodes, while keeping  $KRDBs$  as accurate as possible. Here, we refer to target nodes of synchronization as well-matched nodes. We consider well-matched nodes to possibly be nodes which hold as many identical or similar data items as possible, because consequently they are likely to keep more identical  $KRs$ . This intuition is the same as that of Semantic Overlay Networks (SONs) [23] and P2P file sharing systems with interest-based locality [11]: if nodes share a particular data item, then it is likely that one node will be able to find other semantically similar data items in another one. However, it would be too laborious to check all data items in all other nodes. In addition, it would be difficult to judge remotely whether two data items are identical when their names differ.

To make it easy to find well-matched nodes, we focus on keywords as abstraction of data items. Here, the keywords that can be extracted only from local data items are

called Primary Keywords (PKs) to distinguish them from additional keywords (Secondary Keywords, or SKs), which are added through  $KRDB$  synchronization. In this paper, each node selects the  $N$  best-matched nodes to synchronize with that have the largest number of the shared PKs<sup>†</sup>. Every node periodically searches better-matched nodes and refines the node set for synchronization it keeps. Note that we need not set large value for  $N$  because  $KRs$  between PKs and SKs are diffused in a hop-by-hop manner through  $KRDB$  Synchronization among nodes that share PKs (See 3.2.2).

Though we adopt the above simple metric for node selection in this paper, we could adopt more sophisticated metric such as the number of “weighted” PKs, which takes into account not only the number of shared PKs but also their importance (strength of association) for the nodes. In multimedia content sharing applications we focus on, almost all nodes have interest locality [11], [23]. In such environments, the nodes selected based on the above simple metric and those selected based on the sophisticated metric would be almost the same due to the interest locality. In addition, the node selection mechanism using the simple metric needs less calculation and memory space for additional parameters than the mechanism using the sophisticated metric. Thus, we consider node selection based on the simple metric is sufficient.

In this paper, the way to discover well-matched nodes is simply broadcasting. These broadcasted messages could be merged with query messages. Therefore we expect this method to have no negative effect on scalability. Note that more sophisticated and scalable discovery algorithms, such as multiple random walks [9] would also be applicable.

#### 3.2.2 Synchronization Mechanisms

Subscribing nodes periodically synchronize their  $KRDBs$  with well-matched nodes. Figure 4 shows an overview of the  $KRDB$  synchronization mechanism. First, a node discovers the  $N$  best-matched nodes for  $KRDB$  synchronization in the way described above. The  $KRDB$  synchroniza-

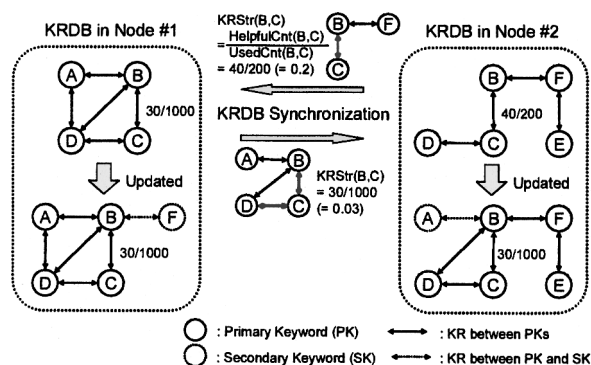


Fig. 4 KRDB synchronization.

<sup>†</sup>The number of selected nodes for  $KRDB$  Synchronization can be less than  $N$  when a node cannot find  $N$  nodes with which share at least one PK.

tion consists of two mechanisms. The first mechanism is KR addition, where a KR between two PKs or a KR between a PK and a SK kept in one node is added to the KRDB in the other node only when one of the two PKs or the PK is shared at those two nodes respectively.

Note that a KR between two SKs in one node is not added to the KRDB in the other node. This limitation can avoid KR explosion: that is, it can prevent successive expansion of newly added SKs.

For example, in Fig. 4, the KRDB in Node #1 is updated by adding two KRs concerning PK B, KR(B, F) and KR(F, B), which are extracted from the KRDB in Node #2, because there exists a shared PK B. In the same way the KRDB in Node #2 is updated by adding four KRs concerning PK B (i.e. KR(B, A), KR(A, B), KR(B, D), KR(D, B), KR(C, D), and KR(D, C)).

KR addition, however, would still have a risk of KR explosion due to adding useless or meaningless KRs with shared PKs. To further suppress KR explosion, we remove such KRs. Specifically, we remove both useless KRs with KRStr smaller than a threshold  $KRStrThresh$ .

Second, there is KRStr modification where KRStr in one node is updated to that in the other node, which would be statistically more accurate, when KRs between two PKs are shared at those two nodes. Here, we consider the accuracy of  $KRStr(k_i, k_j) (= \frac{HelpfulCnt(k_i, k_j)}{UsedCnt(k_i, k_j)})$  statistically increases as  $UsedCnt(k_i, k_j)$  increases. For example, in Fig. 4,  $KRStr(B, C)$  in Node #2 is updated from  $40/200 (= 0.2)$  to  $30/1000 (= 0.03)$  because  $UsedCnt(B, C)$  in Node #1 (= 1000) is larger than that in Node #2 (= 200). Thus each KR radiates from the node that keeps the PK with the largest  $UsedCnt$  through KRDB synchronization.

#### 4. Implementation

We are now developing a prototype semantic P2P search system with query expansion. We have implemented the search algorithms described in Sects. 2 and 3 on LimeWire [25], which is a well known open-source P2P keyword search application. LimeWire is written in Java.

Figure 5 is a screenshot of the prototype (LimeWire with query expansion). The upper window shows the ranked search results. To make clear the difference between the search results of a normal search and those with query expansion, and to help performance evaluation and further development, the prototype simultaneously executes a normal search and a search with query expansion. Note that because a common query routing algorithm is used for these two search mechanisms, no additional traffic is generated. We made it possible to switch the display to show the search results of these two mechanisms. The lower window shows the visualized KRDB. The dotted line denotes the KRs and quadrangles denote keywords.

We are now running the prototype on several PCs in a local area network, and plan to run it globally at several locations through the Internet. We will also try to quantitatively demonstrate the advantages of query expansion though this

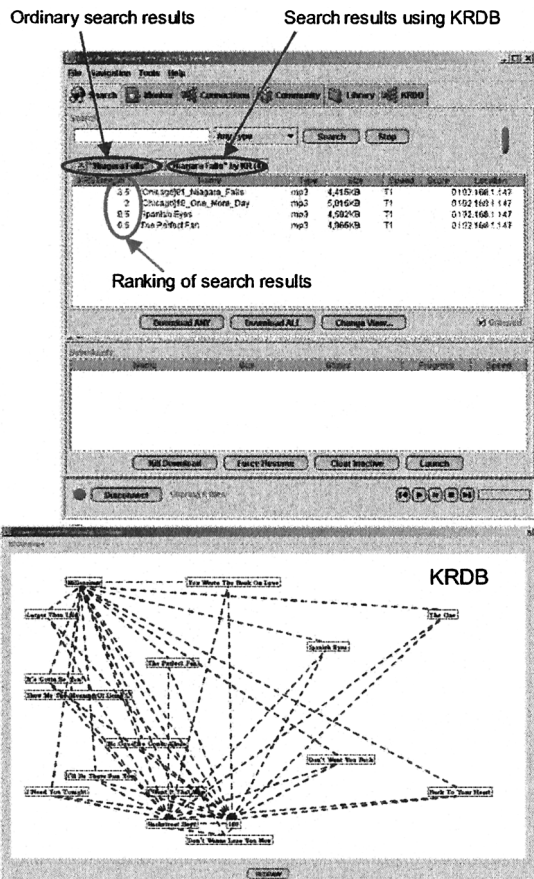


Fig. 5 Implementation on LimeWire.

will be difficult without extensive and continuous use. We discuss the traffic overhead for KRDB updates in Sect. 6.2.

#### 5. Evaluation

In this section, we evaluate our semantic P2P keyword search system through simple analysis of experimental results. We first propose a novel metric *metadata correctness* to evaluate the superiority of our system over traditional non-semantic P2P search systems. Then we show an overview of our experiments and methodology. Finally we analyze experimental results.

##### 5.1 Metric

We propose a novel metric *metadata correctness*  $P_i$  for content  $i$ , which shows the ratio of data items with content  $i$  which are correctly attached all the desired metadata related to content  $i$  to all data items with content  $i$ . In these experiments, we focus on music content, and we define or identify content only by both “artist name” and “song title.” Accordingly we identify a data item only by its content and do not use other metadata such as fine name, encoding parameters, and file size for this purpose.  $P_i$  is defined as follows:

$$P_i = \frac{N_i^{Org}}{N_i^{Sem}}$$

where  $N_i^{Org}$  and  $N_i^{Sem}$  denotes the number of data items which traditional P2P search systems and our one can absolutely find, respectively, when *any* one of desired keywords for content  $i$  is used.

We consider  $P_i$  is useful for evaluating the quantitative superiority of our system over traditional ones when a searcher desires data items with content  $i$ , because  $P_i$  affects *success rate* of keyword search for traditional systems. Here, success rate is defined as the ratio of target data items which can be found by keyword search to all the target data items actually existing in the limited area a query can reach. When one of desired keywords for content  $i$  is used as a query in traditional systems, smaller  $P_i$  results in lower success rate because the possibility that the metadata is attached to the target data items is lower. On the other hand, success rate of our semantic P2P keyword search is always 1.0, because one of desired keywords is expanded to the other desired ones.

## 5.2 Overview of Experiments

We calculate  $P_i$  by analyzing metadata information used for a real P2P file sharing application. We use a free OpenNap [24] client software to obtain metadata information from OpenNap servers, which have the same functionalities as Napster index servers. Though our system is designed based on Gnutella, we believe shared data items in P2P file sharing applications are similar.

Among metadata directly obtained from OpenNap servers, we use only “artist name” and “song title.” We extract another kind of metadata except them from file name, because there is no rule to name data items. For example, we can find some data items with file name “artist name—song title—movie title.mp3.” As a result, at most three keywords can be derived from each data item.

We decide a specific keyword  $x$  used for keyword search and obtain search results from an OpenNap server, from which all metadata are extracted for constructing a *global* KRDB. We refer to this global KRDB as a local KRDB in a node in our experiments. Note that even in distributed environments, each node can construct a part of the global KRDB through KRDB synchronization, so that we believe the same results can be obtained in such environments. From this global KRDB, we select another two keywords  $y$  and  $z$  that have a KR to  $x$ . We calculate  $P_i$  by investigating how many target data items keep all these three keyword.

We did this experiment three times at intervals of at least one day. The number of online users and that of data items were about 3,500–5,000 and 2,100,000–2,500,000, respectively.

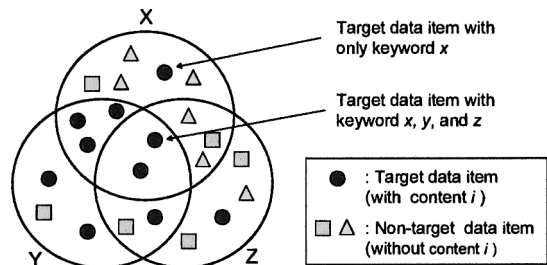


Fig. 6 An example of search results.

## 5.3 Methodology

We describe the methodology using an example of search results shown in Fig. 6.

1. We decide an arbitrary keyword  $x$ , and then obtain a search result (a list of data items) from an OpenNap Server for keyword  $x$ . We refer to the set of data items as  $X$ .
2. We extract metadata from the data items and construct a global KRDB. Then we select keyword  $y$  and  $z$  that have a KR to  $x$ .
3. We obtain a search result from an OpenNap Server for keyword  $y$ . We refer to the set of data items as  $Y$ .
4. We obtain a search result from an OpenNap Server for keyword  $z$ . We refer to the set of data items as  $Z$ .
5. We select a target data item which keeps all the keyword  $x$ ,  $y$ , and  $z$ , and define its content as  $i$ . For example, in Fig. 6, data items represented by a circle are selected because one of them exists in the area  $X \cap Y \cap Z$ . If there exists no satisfying data item, another keyword is selected as  $y$  or  $z$ .
6. We calculate  $P_i$  using the following formula converted from the one defined in Sect. 5.1

$$P_i = \frac{N_i(X_i \cap Y_i \cap Z_i)}{N_i(X_i \cup Y_i \cup Z_i)}$$

where  $X_i$ ,  $Y_i$ , and  $Z_i$  represent a set of target data items with content  $i$  in  $X$ ,  $Y$ , and  $Z$ , respectively, and  $N_i(A)$  denotes the number of target data items with content  $i$  in the set  $A$ . As  $N_i(X_i \cup Y_i \cup Z_i)$  can be converted to  $N_i(X_i \cap Y_i \cap Z_i) + N_i(X_i \cap Y_i) + N_i(Y_i \cap Z_i) + N_i(Z_i \cap X_i) + N_i(X_i \cap \bar{Y}_i \cap \bar{Z}_i) + N_i(\bar{X}_i \cap Y_i \cap \bar{Z}_i) + N_i(\bar{X}_i \cap \bar{Y}_i \cap Z_i)$ ,  $P_i$  can be calculated by counting each of these items.

For example, in Fig. 6, we can count each item as follows:  $N_i(X_i \cap Y_i \cap Z_i) = 2$ ,  $N_i(X_i \cap Y_i) = 3$ ,  $N_i(Y_i \cap \bar{Z}_i) = 1$ ,  $N_i(\bar{Z}_i \cap \bar{X}_i) = 0$ ,  $N_i(X_i \cap \bar{Y}_i \cap \bar{Z}_i) = 1$ ,  $N_i(\bar{X}_i \cap Y_i \cap \bar{Z}_i) = 2$ ,  $N_i(\bar{X}_i \cap \bar{Y}_i \cap Z_i) = 1$ . As a result,  $P_i$  is calculated as follows:  $P_i = \frac{2}{2+3+1+0+1+2+1} = \frac{2}{10} = 0.2$ .

We can derive another metric *metadata correctness* for  $x$ , which represents the ratio of data items with content  $i$  which are attached keyword  $x$  to all data items with content  $i$ .  $P_i(x)$  can be calculated as follows:  $P_i(x) = \frac{N_i(X_i)}{N_i(X_i \cup Y_i \cup Z_i)} = \frac{6}{10} = 0.6$ . In the same way, we can calculate  $P_i(y)$  and  $P_i(z)$

**Table 2** Analysis of experimental search results for content  $i$ .

	Exp. #1	Exp. #2	Exp. #3
$N_i(X_i \cap Y_i \cap Z_i)$	12	11	12
$N_i(X_i \cap Y_i)$	45	38	56
$N_i(Y_i \cap Z_i)$	8	6	9
$N_i(Z_i \cap X_i)$	1	2	2
$N_i(X_i \cap \bar{Y}_i \cap \bar{Z}_i)$	0	0	0
$N_i(X_i \cap Y_i \cap \bar{Z}_i)$	5	8	3
$N_i(X_i \cap \bar{Y}_i \cap Z_i)$	2	6	7
$P_i$	0.16	0.16	0.14
$P_i(x)$	0.80	0.72	0.79
$P_i(y)$	0.96	0.89	0.90
$P_i(z)$	0.32	0.35	0.34

as follows:  $P_i(y) = 0.8$ ,  $P_i(z) = 0.4$ . These results show that our semantic P2P search system performs best in comparison with traditional ones when a searcher issues a query with keyword  $z$ .

5.4 Analysis

Table 2 shows the analytic results of individual experiments. In our experiments, we select the keywords ‘‘Celine Dion,’’ ‘‘My Heart Will Go On,’’ and ‘‘Titanic’’ as  $x$ ,  $y$ , and  $z$ , respectively.

In Table 2,  $P_i$  is at most 0.16. This means that only 16% of overall target data items are given all the three desired keywords ( $x$ ,  $y$ , and  $z$ ). In other words, the percentage of target data items that can be found in traditional P2P search systems when any one of the three keywords is used is only 16%. On the other hand, the semantic P2P search system can find all the target data items if they have at least one desired keyword. Table 2 also shows that  $P_i(z)$  is smallest among  $P_i(x)$ ,  $P_i(y)$ , and  $P_i(z)$ , and our system performs best in comparison with traditional ones when keyword  $z$  is used as a query.

Note that metadata correctness ( $P_i$ ,  $P_i(x)$ ,  $P_i(y)$ , and  $P_i(z)$ ) depends on target content, and the analytic results in this section do not show quantitative characteristics but show only qualitative ones. In our assumed environments where a specific naming standard for data item itself and for its metadata is not defined, the above qualitative characteristics would hold true for other content.

6. Discussion

In this section, we discuss our semantic P2P keyword search system from various aspects. Specifically, we focus on the long-term convergence and overhead of KRDBs through distributed KRDB updates, and interoperability with traditional P2P keyword search systems.

6.1 Convergence of KRDBs

We explain here the long-term convergence of KRDBs through the distributed KRDB updates described in Sect. 3. KRDBs keep KR and the corresponding KRStr for each

KR. Our semantic P2P search system is managed in a fully decentralized manner, so it is important to ensure that KRStr does not oscillate even when nodes frequently join and leave.

In our system, KRStr is updated in two ways in parallel: evaluation feedback (Sect. 3.1) and KRStr modification (Sect. 3.2.2). Recall that KRStr is calculated as  $KRStr = \frac{HelpfulCnt}{UsedCnt}$ . When a node joins a P2P network and constructs its KRDB from local data items, UsedCnt composing KRStr with HelpfulCnt is set as the initial value (currently UsedCntInit = 2), which is small enough to be almost meaningless statistically. As UsedCnt (and HelpfulCnt) is incremented through evaluation feedback processes, it becomes statistically more meaningful. UsedCnt and HelpfulCnt are also modified (updated) through KRStr modification when KRStr with larger UsedCnt is found in another node.

This propagation mechanism of KRStr helps enables fast KRDB updates both for newly joined nodes and the inheritance of removed nodes. Then, in turn, the inherited UsedCnt (and HelpfulCnt) is again gradually incremented through evaluation feedback processes. Thus, for each KR, only the KRStr with the largest UsedCnt are inherited from among the well-matched neighboring nodes as long as the corresponding KR exists in KRDBs in at least one well-matched nodes. In this way, KRDB consistency is preserved among the well-matched nodes, and only KRStr provides a statistically meaningful value.

6.2 Traffic Overhead for KRDB Updates

In this section, we discuss the additional overhead traffic in our system compared to traditional unstructured P2P search systems. Overhead traffic is caused by two KRDB update mechanisms: feedback evaluation and KRDB synchronization. The overhead of feedback evaluation in a search process consists of only KREval messages. As shown in Fig. 3, the number of KREval messages is less than or equal to that of QueryHit messages. Therefore, we believe the overhead of the feedback evaluation does not significantly degrade scalability.

On the other hand, the overhead of KRDB synchronization is difficult to estimate. Instead, we can suppress the overhead at a constant rate by adaptively adjusting parameters such as the synchronization interval ( $T$  [s]) and the number of well-matched nodes for synchronization ( $N$ ) based on traffic measurement at each node. Each node periodically measures the rate of the KRDB synchronization messages to all the received messages including Ping/Pong, QueryHit, Query and KREval messages.

6.3 Interoperability

To gradually deploy a new class of P2P keyword search applications on the Internet, where some P2P applications such as Gnutella are widely used, it is important to provide interoperability with existing P2P keyword search systems. Our

semantic P2P search system is implemented on LimeWire, and provides interoperability with applications implemented with the Gnutella protocol [5]. For simplicity, we refer to a node running our semantic P2P search application as an expandable node, and a node running a traditional Gnutella application as a non-expandable node respectively.

We separately discuss two overlay networks: an overlay for query forwarding that is organized by all the participant nodes, and one for KRDB synchronization that is organized only by expandable nodes. In this section, we discuss the former. The latter was previously discussed in Sect. 3.2.1. Note that if all participant nodes are expandable nodes, we could construct a common overlay network for each use.

When a non-expandable node receives a query generated by an expandable node, the node ignores all the additional fields for semantic search, and deals with the query as an ordinal one: that is, it performs a simple text-match search without query expansion and then forwards it to neighboring nodes.

## 7. Related Work

Works related to ours can be divided into three categories; 1) P2P search using state-of-the-art IR algorithms, 2) P2P search using semantical similarity of content or nodes, and 3) P2P search on distributed databases. The first category is P2P search using state-of-the-art IR algorithms with some extensions suitable for a P2P system. pSearch [19] is an efficient information retrieval system that support semantic-based full-text search like traditional centralized search engines. pSearch focuses on traditional vector space model (VSM) and latent semantic index (LSI), and extends these algorithms to be suitable for DHTs. Thus pSearch is a sophisticated IR system that has both scalability of DHT systems and accuracy of state-of-the-art IR algorithms. pSearch enables semantics search by exploiting P2P LSI (pLSI) and copes with synonym, polysemy, and noise in documents caused by literal matching schemes such as VSM. Our semantic P2P search system differs from pSearch in several points. First, the goal of pSearch is to achieve as accurate keyword search as traditional centralized one. In addition, the target contents of pSearch are general text files (documents). For these reasons, pSearch adopts state-of-the-art IR algorithms. On the other hand, our goal is to improve the possibility of locating content while suppressing results implosion rather than to improve quality of search results. The target contents are non-text files with several keywords such as music files. Second, pSearch needs some pre-computed global statistics such as the dictionary, the inverse document frequency (IDF), and the basis of the semantic space. Our semantic P2P system does not require any such pre-computed or predefined global information.

The second category is P2P search using semantical similarity of content or nodes. This system is based on the following intuition: if nodes share particular content, then it is likely that one node will be able to find other semantically

similar content in another one. Semantic Overlay Networks (SONs) [23] and P2P file sharing systems with interest-based locality [11] are a flexible network organization that improve query performance. With SONs, nodes with semantically similar content are clustered together. Specifically, a node in  $SON_c$  if a significant number of its local document classifies as  $c$ . Documents are classified by probing some existing database such as All Music Guide at allmusic.com, which provides classification hierarchies. In this system, nodes that share similar interests (identical content) create shortcut to one another over existing unstructured overlay networks. Nodes then use these shortcuts to locate content. When shortcuts fail, nodes resort to using the underlying overlay. In these networks, queries are flooded in the same way as Gnutella and our semantic P2P system. These systems focus on neighbor selection based on similarity of content in order to reduce query hops. On the other hand, our semantic P2P system focuses on query expansion based on keyword relationships and our goal is not to reduce query hops, but to enable semantic search in the P2P networks.

NeuroGrid [26] exploits relationship between nodes and keywords. Each node maintains a routing table that associates nodes with keywords. A query is forwarded to other nodes that have previously been associated to the keywords in the query. NeuroGrid focuses on query routing based on relationship between nodes and keywords in order to reduce query hops. Again, our goal is not to reduce query hops, but to enable semantic search.

The third category is P2P search on distributed databases. Edutella [27] is an infrastructure for sharing metadata in RDF (Resource Description Framework) format, which is the W3C metadata standard, in a distributed environment. In this work, a featured RDF query language and an RDF-based metadata model that can support a variety of rich RDF queries are developed. Edutella assumes the target data items are attached complex structured RDF-based metadata. On the other hand, as mentioned in Sect. 1, our target content is multimedia data with several simple unstructured metadata (keywords). Edutella cannot deal with such metadata model. In addition, Edutella assumes relatively reliable and static networks, while our semantic P2P system can essentially adapt to such dynamic environments.

Peer Data Management System (PDMS) [28] and the P2P data mapping system [29] are a decentralized infrastructure for integrating distributed databases or knowledge bases with different formats. In these work, relationships among structured metadata are predefined [28], or relationships between metadata and a data item itself, rather than those among metadata, are used for semantic search [29]. These systems focus on how to mediate existing databases. On the other hand, our semantic P2P system creates and improves such databases through distributed KRDB update mechanisms. Edutella, PDMS, and the P2P data mapping system would fall more into the category of distributed data base systems, rather than P2P search systems which focus on network issues such as query routing or overlay network

topology.

SWAP Metadata Model [30] and H<sup>3</sup> [31] would be the most related work to ours. SWAP and H<sup>3</sup> are ontology-based metadata models which enable semantic search. In these models, each ontology of individual nodes is integrated into a global ontology in a distributed manner in order to support a variety of rich queries. In particular, SWAP develops an RDF-based metadata model suitable for ontology integration and a semantics-based information exchange model, and H<sup>3</sup> proposes a comprehensive framework for peer-based knowledge management sharing and evolution composed by a knowledge infrastructure layer and a communication infrastructure layer. These models, however, focus on synonymy as semantics because complex structured metadata models are predefined. On the other hand, our semantic P2P system focuses on correlation as semantics because, as mentioned in Sect. 1, we do not assume a specific naming and structural standard for metadata.

## 8. Conclusion

In this paper, we have described the basic concept and design of an efficient decentralized P2P search system that supports semantic search through query expansion while retaining the desirable properties of existing unstructured P2P systems (e.g. simplicity and robustness). In our semantic P2P search system, queries are expanded based on KRDBs to improve the possibility of locating a poorly-defined desired data item. We proposed use of a results ranking mechanism to cope with any consequent results implosion. To improve the KRDBs and enhance search performance, we proposed two KRDB update mechanisms: evaluation feedback and KRDB synchronization. We have also described a prototype implementation of our proposed system. Then, we analyzed experimental search results to evaluate our system, and showed qualitative superiority over traditional P2P search systems. Finally, we discussed long-term convergence and overhead of KRDBs through distributed KRDB updates, and interoperability.

The remaining issues include how to protect our system against DoS attacks or malicious users who sends large amount of meaningless queries and intensive performance evaluation. We are now planning to run the prototype at several locations globally through the Internet.

## References

- [1] S. Lawrence and C. Giles, "Searching the World Wide Web," *Science*, vol.280, no.5360, pp.98–100, 1998.
- [2] S. Lawrence and C. Giles, "Accessibility of information on the web," *Nature*, vol.400, pp.107–109, 1999.
- [3] M. Bergman, *The Deep Web: Surfacing Hidden Value*, <http://www.brightplanet.com/deepcontent/>
- [4] Napster, <http://www.napster.com/>
- [5] Gnutella, <http://gnutella.wego.com/>
- [6] Clip2 Distributed Search Services, *The Gnutella Protocol Specification v0.4*, 2000. [http://www9.limewire.com/developer/gnutella\\_protocol\\_0.4.pdf](http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf)
- [7] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella network," *IEEE Internet Computing*, vol.6, no.1, pp.50–57, Jan./Feb. 2002.
- [8] FastTrack, <http://www.fasttrack.nu/>
- [9] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," *Proc. ACM ICS 2002*, June 2002.
- [10] E. Cohen, A. Fiat, and H. Kaplan, "Associative search in peer to peer networks: Harnessing latent semantics," *Proc. IEEE INFOCOM 2003*, April 2003.
- [11] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using Interest-based locality in peer-to-peer systems," *Proc. IEEE INFOCOM 2003*, April 2003.
- [12] P. Ganesan, Q. Sun, and H. Garcia-Molina, "YAPPERS: A peer-to-peer lookup service over arbitrary topology," *Proc. IEEE INFOCOM 2003*, April 2003.
- [13] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P systems scalable," *Proc. ACM SIGCOMM 2003*, Aug. 2003.
- [14] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," *Technical Report, UCB/CSD-01-1141*, April 2000.
- [15] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *Proc. ACM SIGCOMM 2001*, Aug. 2001.
- [16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *Proc. ACM SIGCOMM 2001*, Aug. 2001.
- [17] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Proc. Middleware 2001*, Nov. 2001.
- [18] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, "Complex queries in DHT-based peer-to-peer networks," *Proc. IPTPS 2002*, March 2002.
- [19] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," *Proc. ACM SIGCOMM 2003*, Aug. 2003.
- [20] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," *Proc. Middleware 2003*, June 2003.
- [21] M. Mitra, A. Singhal, and C. Buckley, "Improving automatic query expansion," *Proc. ACM SIGIR '98*, Aug. 1998.
- [22] W. Hersh, S. Price, and L. Donohoe, "Assessing thesaurus-based query expansion using the UMLS Metathesaurus," *Proc. 2000 Annual AMIA Fall Symposium*, 2000.
- [23] A. Crespo and H. Garcia-Molina, *Semantic Overlay Networks*, Submitted for Publication, 2002.
- [24] OpenNap, <http://opennap.sourceforge.net/>
- [25] LimeWire, <http://www.limewire.com/>
- [26] S. Joseph, "NeuroGrid: Semantically routing queries in peer-to-peer networks," *Proc. International Workshop on Peer-to-Peer Computing*, May 2002.
- [27] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "EDUTELLA: A P2P networking infrastructure based on RDF," *Proc. WWW 2002*, May 2002.
- [28] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov, "Schema mediation in peer data management systems," *Proc. ICDE'03*, March 2003.
- [29] A. Kementsietsidis, M. Arenas, and R. Miller, "Mapping data in peer-to-peer systems: Semantics and algorithmic issues," *Proc. ACM SIGMOD 2003*, June 2003.
- [30] J. Broekstra, M. Ehrig, P. Haase, F. Harmelen, A. Kampman, M. Sabou, R. Siebes, S. Staab, H. Stuckenschmidt, and C. Tempich, "A metadata model for semantics-based peer-to-peer systems," *Proc. SemPGRID'03*, May 2003.
- [31] A. Castano, S. Ferrara, S. Montanelli, E. Pagani, and G.P. Rossi, "Ontology-addressable contents in P2P networks," *Proc. SemPGRID'03*, May 2003.



**Kiyohide Nakauchi** received the B.E., M.E., and Dr. Eng. degrees in Information and Communication Engineering from the University of Tokyo, Tokyo, Japan, in 1998, 2000, and 2003, respectively. He is currently a researcher of Internet architecture group of Communications Research Laboratory (CRL), Incorporated Administrative Agency, Tokyo, Japan. His current research interests include peer-to-peer networks, overlay networks, multicast communications.



**Yuichi Ishikawa** received the B.E. and M.E. degrees in Information and Communication Engineering from the University of Tokyo, Tokyo, Japan, in 2000 and 2002, respectively. He is currently working at NEC Corporation. His research interests are in the areas of computer networks, network collaboration, and network services.



**Hiroyuki Morikawa** received the B.E., M.E., and Dr. Eng. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively. He is currently an Associate Professor of the Department of Frontier Informatics at the University of Tokyo. From 1997 to 1998, he stayed in Columbia University as a visiting research associate. His research interests are in the areas of computer networks, ubiquitous networks, mobile computing, and wireless networks. He

serves as Editor of Transactions of Institute of Electronics, Information and Communication Engineers (IEICE) and on the technical program committees of IEEE/ACM conferences and workshops. He is a member of IEEE, ACM, ISOC, IPSJ, and ITE.



**Tomonori Aoyama** received the B.E., M.E. and Dr. Eng. from the University of Tokyo, Tokyo, Japan, in 1967, 1969 and 1991, respectively. Since he joined NTT Public Corporation in 1969, he has been engaged in research and development on communication networks and systems in the Electrical Communication Laboratories. From 1973 to 1974, he stayed in MIT as a visiting scientist to study digital signal processing technology. In 1994, he was appointed to Director of NTT Opto-Electronics Laboratories,

and in 1995 he became Director of NTT Optical Network Systems Laboratories. In 1997, he left NTT, and joined the University of Tokyo. He is currently Professor in the Department of Information and Communication Engineering, Graduate School of Information Science and Technology, the University of Tokyo. His research activities cover the next generation networking technologies from layer 1 (e.g. photonic networks) to higher layers including middleware for network collaboration, P2P routing, mobile networking, and ubiquitous networking. Dr. Aoyama is involved in several governmental projects such as Japan Gigabit Network (JGN) and the Ubiquitous Networking Forum, and in some non-profit organizations and consortiums such as the Photonic Internet Forum (PIF) and the Digital Cinema Consortium (DCC) in which he is serving as Chairman. Dr. Aoyama is IEEE Fellow and was a Members-at-Large of the IEEE ComSoc Board of Governors. He served as President of IEICE Communication Society. He also served Chair of IEEE ComSoc Japan Chapter. He is a member of IPSJ and IEEE.