A Mobile P2P Semantic Information Retrieval System with Effective Updates

Chuan-Ming Liu[*,1], Cheng-Hsien Chen[2], Yen-Lin Chen[1] and Jeng-Haur Wang[1]
[1] Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei City 106 - TAIWAN
[e-mail: cmliu@csie.ntut.edu.tw, ylchen@csie.ntut.edu.tw, jhwang@csie.ntut.edu.tw]
[2] Foxconn Internation Holdings
Tu Cheng Dist., New Taipei City 236 - TAIWAN
[e-mail: c.c.hsien@gmail.com]
*Corresponding author: Chuan-Ming Liu

Received December 26, 2014; revised March 29, 2015; accepted April 19, 2015; published May 31, 2015

Abstract

As the technologies advance, mobile peer-to-peer (MP2P) networks or systems become one of the major ways to share resources and information. On such a system, the information retrieval (IR), including the development of scalable infrastructures for indexing, becomes more complicated due to a huge increase on the amount of information and rapid information change. To keep the systems on MP2P networks more reliable and consistent, the index structures need to be updated frequently. For a semantic IR system, the index structure is even more complicated than a classic IR system and generally has higher update cost. The most well-known indexing technique used in semantic IR systems is Latent Semantic Indexing (LSI), of which the index structure is generated by singular value decomposition (SVD). Although LSI performs well, updating the index structure is not easy and time consuming. In an MP2P environment, which is fully distributed and dynamic, the update becomes more challenging. In this work, we consider how to update the sematic index generated by LSI and keep the index consistent in the whole MP2P network. The proposed *Concept Space Update* (CSU) protocol, based on distributed 2-Phase locking strategy, can effectively achieve the objectives in terms of two measurements: coverage speed and update cost. Using the proposed effective synchronization mechanism with the efficient updates on the SVD, re-computing the whole index on the P2P overlay can be avoided and the consistency can be achieved. Simulated experiments are also performed to validate our analysis on the proposed CSU protocol. The experimental results indicate that CSU is effective on updating the concept space with LSI/SVD index structure in MP2P semantic IR systems.

Keywords: Peer-to-peer networks, Latent semantic indexing, Semantic information retrieval, Distributed two-phase locking

This research was sponsored by research grants from Ministry of Science and Technology, Taiwan under the grant numbers 98-2220-E-027-007, 102-2622-E-027-017-CC3 and 102-2221-E-027-088.

http://dx.doi.org/10.3837/tiis.2015.05.014 **ISSN : 1976-7277**

1. Introduction

As the technologies on communication, computing, internetworking, and software advance, mobile devices now are equipped with more resources and have better computing ability. For example, Qualcomm's Snapdragon 800 series platform has equipped 4 cores with 1.7GHz and Adreno 3XX as well as newer GPU with GPGPU capabilities. Mobile devices are more and more powerful than the old days and become the mobile information and computation centers for users. Using mobile devices to communicate with each other directly thus becomes a trend. **P**eer-**to**-**p**eer (P2P) systems (or networks) are a class of distribution systems enabling users to share resources directly without any intermediates. Each peer (or node) in this paradigm is a service provider as well as a service consumer. In comparison with the Client/Server architecture, the advantages of P2P computing paradigm include: better scalability, sharing and providing information easily among peers, and eliminating the single point failure problem caused by the centralized management in the Client/Server system. From the perspective of the P2P system architecture, using the mobile devices to form a P2P system, **m**obile **P2P** (MP2P) system [1], can be more straightforward and may be a way to alleviate the problem of topology mismatching in P2P systems [2].

In spite of Internet-based cloud services, many people still have no intents to put credential data to the third party service providers. The modern technologies today, as described, create the chance to allow them to efficiently share and index data over mobile devices. For instance, the reviewers in a high credit restaurant review group, where the reviews can only be shared between reviewers internally before publishing reviews to the public. Although one can use a private centralized server to have a data sharing system, the MP2P system, which consists of the mobile devices, provides an alternative, easier and more private way for data sharing by forming the sharing group on the overlay. On such a group, the shared information should be reliable and consistent. The ways about information retrieval (IR), including the development of scalable infrastructures for indexing and searching the data with high accuracy and efficiency, should be reconsidered and become more complicated due to a huge increase on the amount of information and rapid information change on the MP2P system. The index structure thus also needs to be updated frequently. For a semantic IR system, the index structure is even more complicated and generally has a higher update cost than a classic IR system. The most well-known indexing technique used in semantic IR systems is Latent Semantic Indexing (LSI). Although LSI performs well, updating the index structure is not easy and time consuming. In an MP2P environment, which is fully distributed and dynamic, the update becomes more difficult. In this work, we consider how to effectively update the sematic index generated by LSI and keep the index consistent in the whole MP2P network.

Some works and architectures for P2P information retrieval systems have been proposed and most of them use term-index based approaches and propose proper routing protocols to support query execution, especially for short queries [3]. However, term-index based approaches can only support simple retrieval tasks and would be hard to support semantic-aware retrieval tasks. In contrast with term-based indexing, semantic indexing can support many complex retrieval tasks because the information about the whole document is always available. With the semantic information, users can not only retrieve data by concepts but also publish their data into the system data corpus by concepts on P2P systems. Combining P2P networks and semantic information retrieval systems can also improve the scalability and

robustness of the system. Besides, P2P semantic information retrieval systems can grow organically.

In recent years, few works on P2P semantic information retrieval have been proposed, such as pSearch [4], P2PIR [5], Non-orthogonal Binary Decomposition [6], and Semi-Discrete Decomposition [7]. In [8], the generation for globally consistent LSI structures in P2P systems is discussed and a method to construct such LSI structures from distributed corpora is provided. A distributed document search system in P2P network for semantic-based content search is proposed in [9]. In that system, a hierarchical summary indexing structure is used to facilitate semantic-based content search. The paper [10] discusses the possibility about IR on P2P scenarios and proposes a semantic model for P2P overlay network that improves the communication quality between peers. The review [11] also posts the challenge of semantic IR on P2P networks and focuses on the semantic query routing to tailor semantic IR to P2P networks. Then, the author designs a semi-structure semantic overlay for IR in [12]. The approach of pSearch [4] leverages the semantic index and the emerging distributed hash tables (DHT) to enable semantic information retrieval operations on P2P networks. The pSearch uses Latent Semantic Index (LSI) [13][14][15] to generate the semantic index, also called *concept space*. The generated index is static and if some documents with new concepts are added into the data corpus, the whole index will be recomputed, thus resulting in inefficient updates. P2PIR [5] is based on the pSearch and tries to overcome the weakness of the static index generation in pSearch. Based on pSearch, P2PIR uses a different concept indexing approach for constructing the concept space, but the indexing power is not as powerful as the one used in pSearch. Non-orthogonal Binary Decomposition [6] and Semi-Discrete Decomposition [7] also use other index generation techniques and the resulting sematic indices under-perform the concept space used in pSearch.

The above existing works do not address the problems of sematic index update and consistency. In this work, we focus on such problems in MP2P information sharing systems with Latent Semantic Indexing. LSI is a well-known concept space (*i.e.*, document semantics) generation technique and has been used in many modern search engines, like Google and Yahoo. Due to the concept space, the underlying P2P system we consider is the Content Addressable Network (CAN) [16], which has the spatial properties that can tailor the sematic index to the MP2P system, as pSearch does. The LSI uses singular value decomposition (SVD) [17] to derive the semantic index. In order to update the index, the computation involves the matrix multiplications. On the MP2P netowrk, which is distributed and dynamic, the index at each node should be the same. If there are more than one updates, the computation at each node for updating the index should be the same. In other words, the order of the multiplied matrices at each node should be the same since the matrix product is not commutative. This makes the update process more challenging. We thus propose the Concept Space Update (CSU) approach to provide an MP2P semantic information retrieval system, which not only generates the sematic index using LSI but also allows dynamic updates for the concept space over the MP2P network. Once one or more documents have been published, the concept space should be updated to keep the precision of the query results. Two objectives for the proposed system: (1) the index in the MP2P system should be always consistent and (2) the process for updates should be efficient and avoid re-computing the whole index.

As pSearch does, in our design, each peer node will have a copy of the sematic index. We consider the two-phase locking (2PL) scheme used in distributed transactional systems [18] for alleviating the impact of node mobility and frequently disconnection in the MP2P systems and use the SVD-updating proposed in [15][19] to overcome the problem of static index mentioned earlier. With the two-phase locking scheme, the semantic indexing structure in the

whole overlay can be consistently updated and dynamic semantic index generation can be also achieved. The contributions of this paper include:

(1) the development of the mechanism enables the LSI index updates over MP2P networks, which is fully distributed and dynamic;
(2) the consistency of index structure is guaranteed; and
(3) the improvement of the feasibility of LSI-based P2P semantic information retrieval applications.

In order to evaluate the proposed approach, two measurements, *coverage speed* and *update cost*, are considered. The coverage speed is the average time for an update to be spread on the whole network and the update cost is the total number of messages generated for update messages. The analytical and simulated results for the performance on these measurements are also given.

The organization of the rest of this paper is as follows. In Section 2, we introduce some preliminaries and brief the related works. Section 3 gives the proposed CSU approach and the related mathematical analysis is shown in Section4. In Section5, we present the results of our simulation experiments and compare the experimental result with the analytical ones. Last, Section 6 concludes this paper.

2. Preliminaries

In this section, we give some preliminaries and review the mechanisms and concepts used in this work. We first shortly introduce the P2P systems, including Content Addressable Network (CAN) [16]. Then, the indexing models for information retrieval are presented.

2.1 P2P Networks

P2P networks can be classified into two categories, structured and unstructured ones, according to the network organization. An unstructured P2P network is easily formed when the links among nodes are established arbitrarily. On such a system, the queries have to be flooded through the whole network in order to derive the results. Such a query process may cause a high cost and degrade the performance. In contrast, structured P2P networks are organized into a pre-designed structure. The structure is maintained during the P2P network works. The queries thus can be processed efficiently according to the properties of the pre-defined structure. Most of the structured P2P networks employ *distributed hash tables* (DHTs) to manage the distribution of data and support efficient location service. The P2P systems using DHTs organize the nodes in an *overlay network*, which is an abstract and logical network built on the top of a physical network. DHTs characteristically support the following properties: decentralization, scalability, fault tolerance, and load balancing. The P2P system based on DHTs includes Chord [20], CAN, Pastry [21], Tapestry [22], and Kademlia [23].

The content addressable network (CAN) is one of the first proposed DHT systems and uses a virtual *d*-dimensional Cartesian coordinate space for indexing. This coordinate space is completely logical and bears no relation to any physical coordinate system. At any time instance, the coordinate space is dynamically partitioned according to the nodes in the system such that every node owns its individual zone within the overall space. **Fig. 1(a)** shows a 2-dimensional unit coordinate space partitioned with 5 CAN nodes. Once a lookup is issued, it will be hashed into the virtual *d*-dimensional Cartesian coordinate space first. Such a hashed value is called a query point in the space and the query is routed to the node which owns a region containing the query point. Each node in CAN maintains a set of links to its neighbors

including their regions and passes a query to a node whose region containing or being closer to the query point. An example of a lookup for (x, y) issued at node 1 is shown in **Fig. 1 (b)**. Since the space is managed with corresponding to all the existing nodes in the system, the node's join and departure will cause some divided zones to be merged or separated.

Due to the spatial properties in CAN, it is suitable for spatial-based or vector-based information retrieval models. The work of pSearch is the first semantic-enabled P2P information retrieval framework on CAN. In pSearch, the concept space generated via SVD is used as the coordinate space by the underlying CAN. This indicates that LSI [13] is applied to index documents. However, pSearch does not consider how to keep the indices consistent in P2P systems.

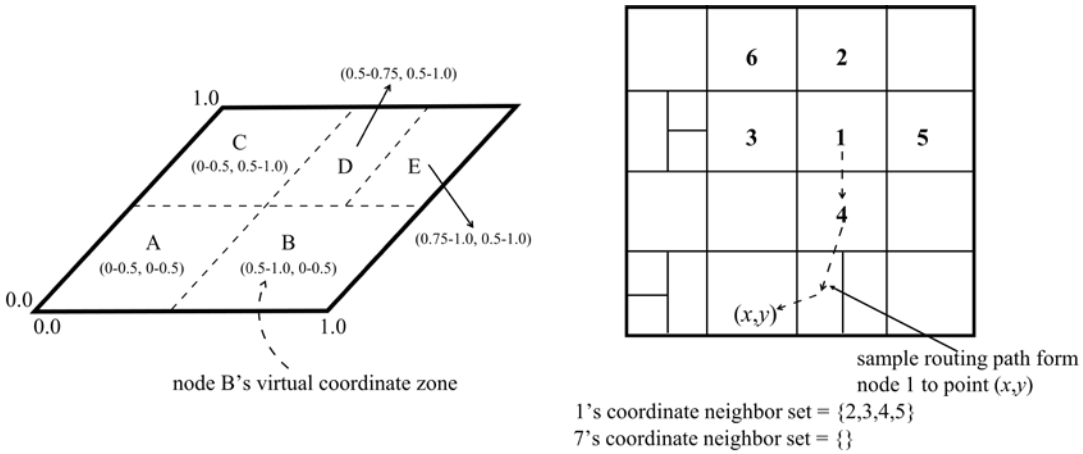

Fig. 1. (a) A content addressable network having 5 nodes within a 2-dimensional space where each node owns a partitioned zone and (b) a lookup process in CAN where the dashed lines are the routing path from node 1 to the query point (x, y)

2.2 Semantic Information Retrieval and LSI

In Information Retrieval (IR), some term-based indexing models were proposed and one of the famous and fundamental indexing models is the Vector Space Model (VSM). In VSM, documents and queries are modeled as vectors of terms, and each dimension corresponds to a separate term. Then, each term is given a weight to represent the importance and relations between documents or in the applications. VSM allows partially matches between queries and documents and is different from the Boolean Model which only produces exact matches. However, for the semantic-aware retrieval tasks, where *polysomies* and *synonymies* may occurs, VSM may not perform well. Polysomies and synonymies will degrade the recall and precision of query results.

Semantic Information Retrieval is evoluted from IR. To deal with the polysomies and synonymies in Semantic IR, some indexing approaches using concepts are proposed. LSI is the mostly used approach and can be viewed as an extension of VSM having the concepts in documents considered. LSI maps each document and query vector into a lower dimensional space associated with concepts. The space is the *concept space* or *semantic space*. In order to find out the patterns for words and documents, LSI uses Singular Value Decomposition (SVD). After having the patterns, LSI can index the documents and terms with concepts. In general,

there are three steps in LSI to construct the concept space: (1) constructing a term-document matrix; (2) performing an SVD on the matrix; and (3) using the matrix to identify the concepts contained in the text. Since LSI and VSM use a multi-dimensional space to index the objects, the similarity of objects can be computed in the same way.

LSI using SVD to project the term-document matrix into a lower dimensional space can benefit from merging term-based dimensions with the same concept in VSM space[1][13][14]. It has the drawback of being computationally expensive. In fact, even using the most advanced numerical linear algebra techniques, the majority of the processing time in LSI is taken up with computing the concept space by SVD [24][25]. In dynamic document publishing environments, the term-document matrix undergoes with time axis. Recomputing the concept space via SVD thus can be prohibitively expensive. Although using the folding-in method to update the semantic space is much faster and cheaper than recomputing, it may result in a significant degradation of the indexing power of the concept space. A more accurate approach, *SVD-Updating*, is then proposed to update the SVD [19]. SVD-Updating is more complicated and costly than the folding-in approach but needs much less computation than recomputing the whole concept space. Moreover, in the absence of round-off error, SVD-updating gives the exact SVD of modified term-document matrix. In SVD-Updating, the orthogonality of concept space is pulled and analyzed via QR-decomposition first. The new document will be processed by the orthogonality and the pulled orthogonality will also be adjusted by new document using a more light-weight SVD process. Finally, the semantic space will be altered by adjusting the orthogonality and the new document will be indexed. In comparison with the recomputing the term-document matrix in SVD, the amount of computation in SVD-Updating is relatively small due to some small intermediate matrices, even though both QR-decomposition and SVD calculation are involved.

2.3 Two-Phase Locking Scheme

Two-Phase locking scheme is originally used in the Centralized Transactional System (CTS). As the advancement of internetworking, many Distributed Two-Phase Locking Protocols (D2PLs) are proposed [18] for the Distributed Transactional System (DTS). The goal of D2PL is to schedule the transactions and avoid conflicting locks being held simultaneously. Obviously, D2PLs would consume more resources than centralized 2PLs due to the extra communication cost between the distributed sites. P2P Transaction System (P2PTS) is a variation of DTS. In comparison with DTS, the distribution behavior in P2PTS is more free and less-controlled while data and resources are held and managed by peers(users) itself. In addition, due to the dynamic nature of P2P networks, the join and departure of nodes are also non-predictable. Building a P2P transactional system thus should pay more efforts on overcoming the dynamic properties of the overlay networks. In [26], a transactional system on the top of a structured P2P network is proposed where the concept of *probation* is introduced. With the probation, the deadlock can be avoided. Our proposed CSU protocol will use the D2PL mechanism and the details will be introduced in the next section.

3. Concept Space Update (CSU) Approach

The proposed CSU protocol is based on the distributed two-phase locking scheme. There are two roles of nodes in CSU when there are new documents to be published: *update participant* (UP) and *update requester* (UR). The update participants are the nodes who receive update requests and update their indices locally as well as forward the requests to their neighbors. The update requesters are the nodes that initiate updates. They will update their indices locally and

then send the update requests to the network. To forward a message in SCU, we consider the *reverse path forwarding* strategy [27], that is a flooding-based routing protocol and appropriate for CAN networks. We use the state diagram in **Fig. 2** to describe the processes of UP and UR, respectively.

To have an effective update, a UR will claim to lock all the nodes and then update the concept space accordingly. When a node has some new documents to be published, it becomes a UR and the state diagram of a UR is shown in **Fig. 2**. The UR first converts the new documents to a corresponding document vector and then moves to **UpdateInit** state. In **UpdateInit** state, the UR initiates a lock request to lock nodes for the following update. The lock request has the node ID, update priority, and the value of time-to-live (TTL). Once the lock request has been setup, the *lock timer* starts and the UR sends the lock request to the network and goes to **WaitForLock** state to wait for a successful lock.

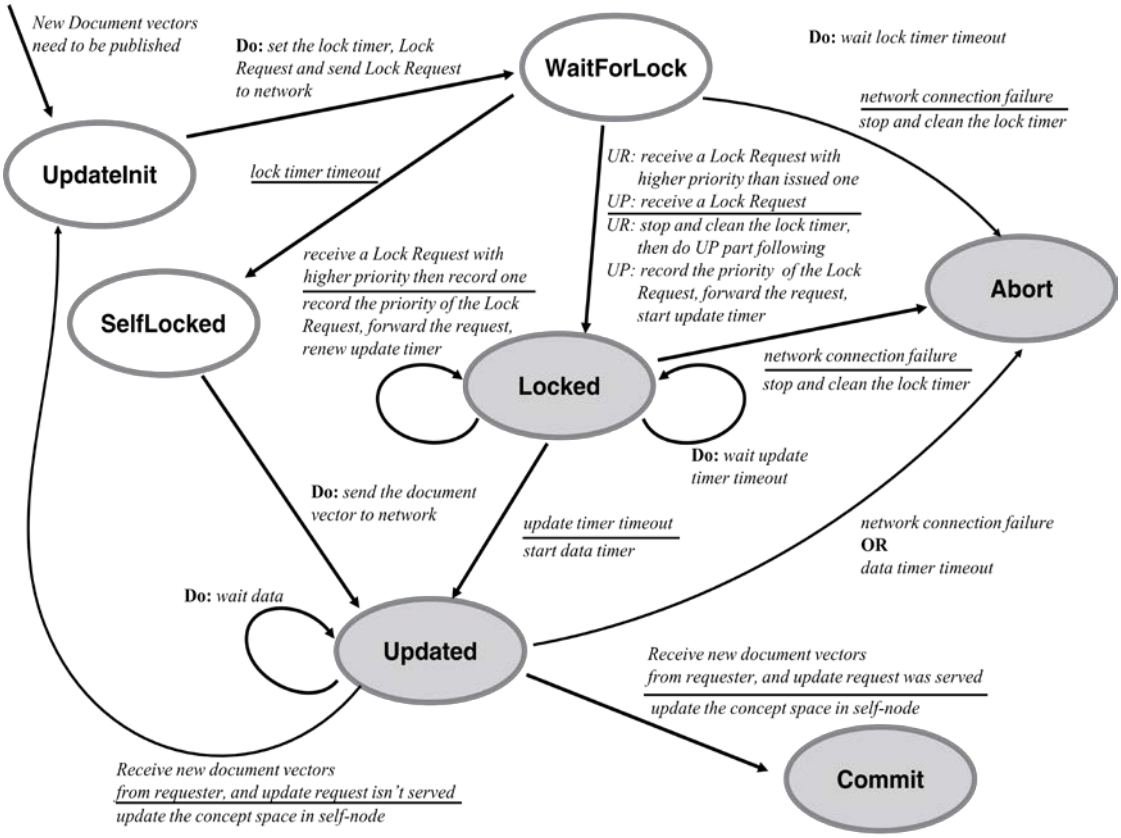

Fig. 2. The state diagram for an Update Requester where the shaded states are the part for an Update Participant

If a UR receives other lock request with a higher priority, the UR cancels its request and moves to **Locked** state to participate the update process from the other node. In this case, the UR becomes a UP. Suppose that no other request with a higher priority is received during TTL. The *lock timer* will trigger the UR to transit into **SelfLocked** state. This indicates that the whole network is locked and ready for an update. If a UR goes to the **SelfLocked** state, there should be only one UR in the network. The UR then sends the updated document vectors to the network and moves to **Update** state, which is the main process in a UP. When the lock process

fails, the UR will restart the whole process after the request with a higher priority is committed.

The update process of a UP is easier than the one of a UR since a UR needs to deal with the self-issued update. The shaded states in **Fig. 2** present the state diagram of a UP. When a UP receives a lock request (LR) issued by a UR, it starts the *update timer* using the *lock timer* in the lock request and moves into **Locked** state. In a mobile environment, due to the mobility and frequent disconnection, locking all the nodes may lengthen the time for an update and sometimes is even impossible. Hence, when the update timer expired, some neighbors of a UP may not be in **Locked** state. To ensure all the active nodes are locked within a specific time period, we use the knockout mechanism, which is originated from the underlying CAN protocol, to knock down all the neighbors not in **Locked** state. The nodes that have been knocked out then can reconnect to the network after the update and synchronize the latest semantic index with one of its neighbors. By paying little extra network traffic, using the designed lock timer and knockout mechanism can effectively alleviate the impact of node mobility and frequent disconnection on the time for an update in a distributed and mobile environment.

The UP then sets up the *data timer* to wait for the document vector which should be updated into concept space and moves to **Update** state. In **Update** state, if the updated document vectors received, the UP updates the concept space locally via SVD-Updating and commits the request. During the process, if the network connection fails or the data timer has been expired, the whole update process will abort.

In mobile environments, the updates may be issued from different nodes at the same time. In this case, the request conflicts will occur. A node who detects the conflict will notify the UR that issues the update and the notified UR will reissued the request based on some confliction strategies. We consider two strategies for study. The first strategy is *contention-based*, where the conflicted update request will be re-issued after a random time period. The other strategy uses a priority on the request. The requests with a higher priority will be stored if a UR gets the conflict notification from the other URs. The UR will check the requests it stores to see when it can re-issue the request. To differentiate the second strategy from the first one, we refer to this strategy as *contention-avoid* strategy. The priority assigned to an update request should be coordinated. A newer request should have a lower priority than the older ones. In the proposed CSU, we use the timestamp as the priority of a request. The lower value has a higher priority. The timestamp is easy to implement and fits the considered environment. In order to avoid the case that multiple requests have an identical timestamp, we append the node id to the timestamp for breaking ties.

3.1 Values of Timers

In CSU, the timers play important roles to decide the time for state transition. Correct values make the CSU work in regular and good performance; otherwise, CSU is boggy. Hence, how to compute the correct values for timers becomes a critical issue in CSU.

There are three types of timers in CSU: *lock timer*, *update timer*, and *data time*r. The lock timer is initiated and sent by a UR. It is used to lock the whole overlay nodes such that the afterward update process can perform consistently. If the lock time is long, more nodes can be involved in CSU. However, if the lock time is set too long, say larger than the time needed for locking the whole overlay, the CSU performance can also be degenerated. The update timer depends on the lock timer in a UP. After receiving the lock timer, a UP starts the update timer and the update timer has the same expiration time with the received lock timer. After the

expiration time of the lock timer or update timer, the node (UR/UP) moves to **Update** state and the data timer starts. The time period of the data timer is also derived from the lock timer and used to wait for the document vector sent by the UR to arrive.

The values of timers in CSU are important but simple to be computed. By the CAN properties, the network topology, distribution, and size can be retrieved trivially. Therefore, we can estimate the suitable values for the timers in CSU with the average transmission delay of the overlay network.

3.2 Node Departure and Recovery

The node departure from an MP2P network and failure are processed in the same way in CSU, but the node behavior will depend on the node types. If the departure node is a UR of current updated request, the request will be aborted and all the acquired locks are released. In the other case, the node would be simply skipped in the update process.

To keep the consistency of the concept spaces in CSU, nodes departed from the network must restore the space back if it reconnects. In CSU, nodes reconnecting and joining will be processed in the same way. When a node joins to the system network, it first synchronizes the concept space with its bootstrapping node such that the concept space on each node can be consistent. If the reconnected node is a UR, it must synchronize the concept space before re-issuing its uncompleted update request.

4. Analysis on the CSU

In the considered environment, the updates occur dynamically. In order to evaluate the proposed CSU, we consider two measurements: *coverage speed* and *update cost*. The coverage speed is to measure the average time for an update request to update the network throughout. The other measurement is to see how much network traffic an update brings and check the total number of messages generated by updates.

4.1 Coverage Speed

Recall that we use reverse path forwarding strategy for message propagation in CSU and the underlying P2P network is CAN. Suppose there are n nodes in the considered CAN network N, of which the dimension is d. Let $T_{d,v}$ denote the update time for a request issued by a node $v \in N$. Then, the average time for an update request to update the whole network is

$$T_{d,ave} = \frac{\sum_{v \in N} T_{d,v}}{n} \tag{1}$$

We use a CAN network of dimension 2 to show how to derive $T_{2,ave}$. The result can be extended to an arbitrary dimension d and $T_{d,ave}$ can be derived accordingly. Consider the 2D CAN network in **Fig. 3** and d=2. Suppose that the original (0, 0) is located at the center and the whole space is partitioned into 4(=d^2) quadrants. We consider one of the quadrants first and the others can be done similarly. Let node v is located at (i, j) in the first quadrant. Without loss of generality, $T_{d,v}$ can be derived by considering that node v needs to send its message to the node farthest from v. That is,

$$T_{d,v} = (i + j + \frac{\sqrt{n}}{2} + \frac{\sqrt{n}}{2}). \tag{2}$$

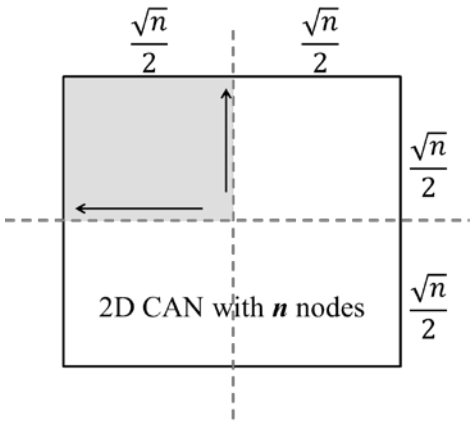

Fig. 3. Node organization in a 2D CAN network

So, the total update time T_q for all the nodes in the quadrant is

$$\begin{aligned} T_q &= \sum_{i=1}^{\frac{\sqrt{n}}{2}} \sum_{j=1}^{\frac{\sqrt{n}}{2}} (i + j + \frac{\sqrt{n}}{2} + \frac{\sqrt{n}}{2}) \\ &= \sum_{i=1}^{\frac{\sqrt{n}}{2}} \sum_{j=1}^{\frac{\sqrt{n}}{2}} (i + j) + \sum_{i=1}^{\frac{\sqrt{n}}{2}} \sum_{j=1}^{\frac{\sqrt{n}}{2}} \sqrt{n} \\ &= \frac{n}{4}(1 + \frac{3}{2}\sqrt{n}). \end{aligned} \tag{3}$$

The average time for a node to update a message in the whole network thus can be derived as

$$\begin{aligned} T_{2,ave} &= \omega \times \frac{(2^d \times T_q)}{n} \\ &= \omega \times (\frac{3\sqrt{n}}{2} + 1), \end{aligned} \tag{4}$$

where ω is the average transmission time on the overlay network and d=2. In the calculation, it is not difficult to observe that $T_{2,ave}$ grows in the order of $\sqrt{n}$.

With the similar approach, one can derive $T_{d,ave}$ for a d-dimension CAN network of size n and

$$T_{d,ave} = \omega \times (2^d \times \sum_{c_1=1}^{\frac{\sqrt[d]{n}}{2}} \sum_{c_2=1}^{\frac{\sqrt[d]{n}}{2}} \cdots \sum_{c_d=1}^{\frac{\sqrt[d]{n}}{2}} ((d \times \frac{\sqrt[d]{n}}{2}) + c_1 + c_2 + \cdots + c_d)) / n, \tag{5}$$

where c_i is the absolute value of the coordinate on the i-th dimension and i =1, …, d. To

compute $T_{d,ave}$, let $m = \frac{\sqrt[d]{n}}{2}$ and then $\sum_{c_i=1}^{\frac{\sqrt[d]{n}}{2}} c_i = \sum_{c_i=1}^{m} c_i = \frac{m(m+1)}{2}$. Equation (5) thus can be

$$\begin{aligned} T_{d,ave} &= \omega \times (2^d \times \sum_{c_1=1}^{m}\sum_{c_2=1}^{m}\cdots\sum_{c_d=1}^{m}((d\times m)+c_1+c_2+\cdots+c_d))/n \\ &= \omega \times (2^d \times ((d\times m^{d+1}) + \sum_{c_1=1}^{m}\sum_{c_2=1}^{m}\cdots\sum_{c_{d-1}=1}^{m}(m\times(c_1+c_2+\cdots+c_{d-1}) + \frac{m(m+1)}{2})))/n \\ &= \omega \times (2^d \times ((d\times m^{d+1}) + \sum_{c_1=1}^{m}\sum_{c_2=1}^{m}\cdots\sum_{c_{d-2}=1}^{m}(m^2\times(c_1+\cdots+c_{d-2}) + (2\times m\times\frac{m(m+1)}{2}))))/n \\ &\vdots \\ &= \omega \times (2^d \times ((d\times m^{d+1}) + (\frac{d}{2}\times m^d \times m(m+1))))/n. \end{aligned} \tag{6}$$

Since $m^d = \frac{n}{2^d}$,

$$\begin{aligned} T_{d,ave} &= \frac{d}{2}\times\omega\times(m+1) + \omega\times d\times m \\ &= \frac{3\times m+1}{2} \\ &= \omega\times d\times\frac{3\times\sqrt[d]{n}+2}{4}. \end{aligned} \tag{7}$$

Therefore, $T_{d,ave}$ is in the order of O($\sqrt[d]{n}$).

4.2 Update Cost

In CSU, we consider the overlay network and use the idea of reverse path forwarding [24], which floods the messages in the physical networks. In other words, the number of messages flood on the overlay is measured. However, the analytical results show that reversed path forwarding will not cause the message explosion in CAN network (*i.e.*, the overlay), and the experimental result in Section 5 also confirms this. Our analysis starts with a single request and then we extend the analysis to multiple update requests.

On the considered *d*-dimensional CAN network *N* of size *n*, let *V* and *E* denote the set of nodes and the set of edges in *N*, respectively. We use $M_{d,sin}$ to denote the maximum number of messages produced in the whole network for a single update. Note that, for a *d*-dimensional CAN network, the maximum number of neighbor links for a single node is 2*d*. Since the messages are routed using flooding, at each node, a request will be sent on the neighbor links and the maximum number of update messages sent from a node will be 2*d*. Besides, each edge in *N* will carry at most two messages for an update request. When *d*=1, all the nodes are in a line and the maximum number of messages for an update will be $2\times1\times(n-1) = \mathrm{O}(n)$. For a 2D CAN network of size *n* as **Fig. 3**, there will be $2\times\sqrt{n}\times(\sqrt{n}-1)$ edges. Hence, the maximum number of messages for an update request is

$$M_{2,sin} = 2\times(2\times\sqrt{n}\times(\sqrt{n}-1)) = 4\times(n-\sqrt{n}) = \mathrm{O}(n). \tag{8}$$

Furthermore, $M_{2,sin} = \Omega(n)$, since at least n forward actions needed for a request to cover the whole network. So, $M_{2,sin} = \Theta(n)$ in asymptotic view. This result can be extended to a d-dimensional CAN network, d>2, as

$$M_{d,sin} = 2 \times (d \times (\sqrt[d]{n})^{d-1} \times (\sqrt[d]{n} - 1)) = 2 \times d \times (n - \sqrt[d]{n}) = \mathrm{O}(n). \tag{9}$$

Again, $M_{d,sin} = \Theta(n)$ since $M_{d,sin} = \Omega(n)$ with the same reason as above.

We now consider the case that multiple update requests are issued simultaneously and the contention between requests may occur. In our proposed protocol, the request having the highest priority will win the contention and the others with a lower priority will fail. The failure nodes then compete with each other in the later rounds. Suppose that there are k requests issued simultaneously. The request with the highest priority produces $M_{d,sin}$ messages. The request with second priority will have at most $M_{d,sin} + \frac{1}{2} \times M_{d,sin} = (1 + \frac{1}{2}) \times M_{d,sin}$ messages. It occurs when two requests issued at the two ends of a diagonal simultaneously due to the topology of CAN network. In turns, the request with the ith priority, $1 \le i \le k$, will produce at most $(1 + \frac{1}{2} + \cdots + \frac{1}{i}) \times M_{d,sin}$ messages. Hence, the maximum number of messages for k requests in a contention environment, $M_{d,mul}$, can be calculated as

$$\begin{aligned} M_{d,mul} &= M_{d,sin} + (1 + \frac{1}{2}) \times M_{d,sin} + (1 + \frac{1}{2} + \frac{1}{3}) \times M_{d,sin} + \cdots + (1 + \frac{1}{2} + \cdots + \frac{1}{k}) \times M_{d,sin} \\ &= (k + (k-1) \times \frac{1}{2} + \cdots + 2 \times \frac{1}{k-1} + \frac{1}{k}) \times M_{d,sin} \end{aligned} \tag{10}$$

W.L.O.G., we suppose k is power of 2 and $k=2^p$, for some integer $p \ge 0$. Equation (10) can be further considered as

$$\begin{aligned} M_{d,mul} &= (k + (k-1) \times \frac{1}{2} + \cdots + 2 \times \frac{1}{k-1} + \frac{1}{k}) \times M_{d,sin} \\ &\le M_{d,sin} \times (k + [(k-1) \times \frac{1}{2} + (k-1) \times \frac{1}{2}] \\ &\quad + [(k-3) \times \frac{1}{4} + (k-3) \times \frac{1}{4} + (k-3) \times \frac{1}{4} + (k-3) \times \frac{1}{4}] \\ &\quad + \cdots + [(k - (2^p - 1)) \times \frac{1}{2^p} + \cdots + (k - (2^p - 1)) \times \frac{1}{2^p}] \\ &= M_{d,sin} \times (k + (k-1) + (k-3) + \cdots + (k - (2^p - 1))) \\ &= M_{d,sin} \times (k + p \times k - \sum_{i=1}^{p} (2^p - 1)) \\ &\le M_{d,sin} \times k \log k. \end{aligned} \tag{11}$$

Each request in contention environments thus should spend at most $\log k \times M_{d,sin}$ messages to make the request served. As for the contention avoid strategy, since the collisions are avoided

for the updates, we expect the growth of the number of messages needed for multiple requests should be linearly proportional to the number of simultaneously issued requests.

5. Simulation Experiments

To verify the performance of our proposed mechanisms and analyzed results, the simulation experiments are performed. The parameters used in our self-built simulator are shown in **Table 1**. The number of nodes in the network is ranged from 200 to 2000 and the dimension of the CAN network is from 2 to 5. The transmission delay is set to be 1 to 100 time units. The results are the average of 100 randomly generated update requests. Furthermore, the number of simultaneous requests is from 1 to 500. We first present the reason why the *reverse path flooding* for message propagation is used in the proposed CSU and then discuss the results about the coverage speed and update cost.

Table 1. Experiment Setting

Name	Values	Default
Network Size	200,400, …, 2000	2000
Dimension of CAN	2,3,4,5	2
Transmission Delay	1-100	50

To show the adaptability of reverse path flooding in Concept Space Updating, we compare the reverse path flooding with *random walk traversal*. In random walk traversal, messages will be randomly forwarded and multiple walkers are used to enhance the performance. In our simulation, the coverage speed is measured in the time ticks (units). **Fig. 4** shows the coverage speeds of the considered reverse path flooding and random walks with 1, 4, and 16 walkers respectively in a 2D CAN network. The reverse path flooding performs much better than random walks, even when there are 16 walkers in the traversal, for coverage speed. This demonstrates that adapting reverse path flooding in CSU is much better than using random walk traversal for message propagation.

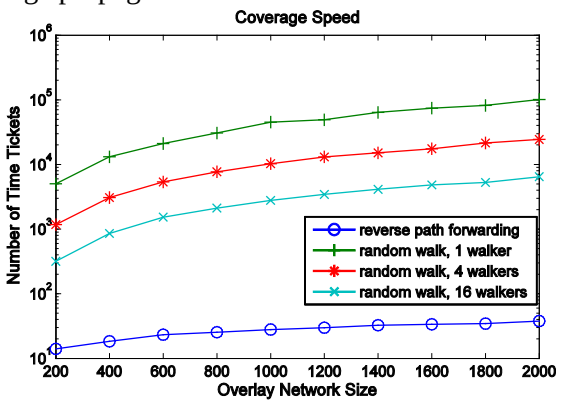

Fig. 4. Coverage speeds of different message propagation methods

We then present the analytical coverage speed and the experimental one in **Fig. 5** for different 2D CAN network sizes. Basically, the experimental results meet the theoretical analysis. Since the request traversed in network will follow the path with the smallest delay but not the path with minimized hops according to reverse path flooding strategy, the coverage

speed can be further shorter in practice. **Fig. 6** is an example showing the actual delay versus the minimum number of hops about message delivery. In this example, the message from *A* to *D* will take the path *A*-*B*-*C*-*D* instead of *A*-*D* directly due to a short delay. As a result, the experimental results are about 20% to 30% less than the analytical one in the coverage speed. Furthermore, for a CAN network of dimension $d \geq 2$, the experimental result confirms the analytical result, where the coverage speed changes in proportion to with the dth root of the network size, *i.e.*, $T_{d,ave}$ is in the order of $O(\sqrt[d]{n})$. Please refer to **Fig. 7**. As the dimension increases, the distance between two nodes is shorter; thus, leading to less time to spread the requests to the whole network.

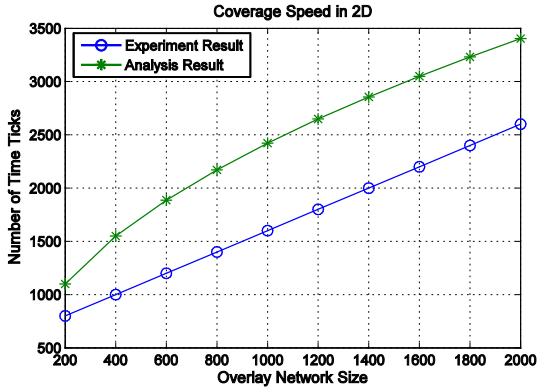

Fig. 5. Coverage speeds in different 2D CAN network sizes

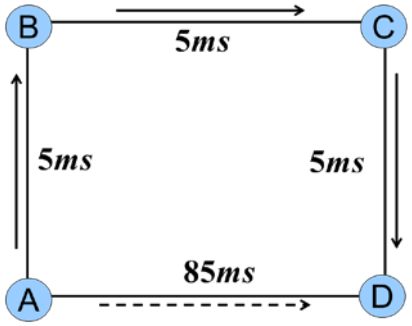

Fig. 6. An example for message routing using actual transmission delay instead of the minimum number of hops

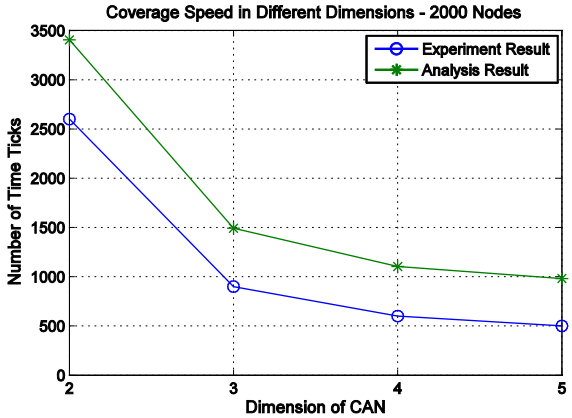

Fig. 7. Coverage speed in different dimensions of CAN for experimental and analytical results

To evaluate the update cost, we measure the total number of messages issued on all the nodes for update requests. **Fig. 8** shows the total number of messages associated with an update request in no contention situation. The number of messages increases as the size of network becomes larger and the update cost grows linearly with the network size. The results meet the analysis in Section 4. We also plot the curve for the twofold of the number of edges to verify the analytical results.

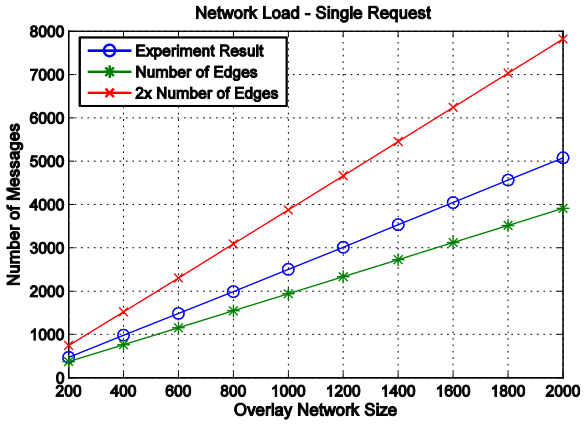

Fig. 8. Update cost for a single request where no contention is performed

For the case of multiple requests issued simultaneously, according to our analysis, when there are k requests, the total number of messages is at most klog k times the maximum number of messages produced in the whole network for a single update, Md,sin. In average, the update cost of each request will be at most log k times the update cost of a single update. **Fig. 9** presents the growth of update cost as the number of simultaneous requests increases. The results about the contention-based strategy are consistent with our analysis and the contention-avoid strategy will have a much lower update cost. Besides, the contention-avoid strategy can be expected to have a linear growth with the number of request messages on update cost.

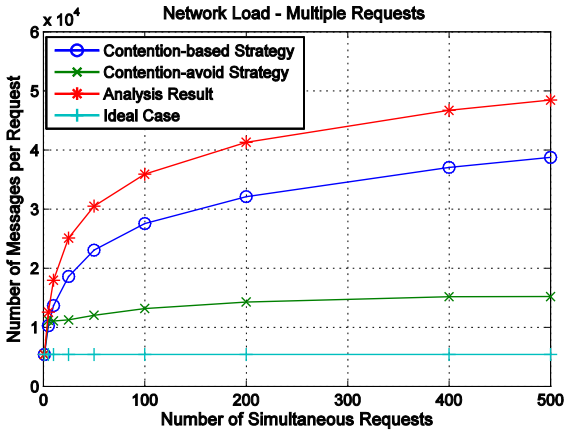

Fig. 9. Update cost of multiple requests where contention occurs

5. Conclusions

Using mobile devices to communicate with each other directly has been a trend for information sharing and can form an MP2P system. In this paper, the indexing issue of semantic information retrieval system on MP2P networks is addressed. We propose the CSU protocol which can ensure that the whole concept space in the overlay can be kept and updated consistently. The CSU combines the SVD-Updating and D2PL, where the incremental update is enabled and the concept space re-computation is avoided. The node departure and recovery in MP2P networks are also discussed. Furthermore, regarding the update conflicts, we provide contention-based and contention-avoid strategies. As a result, the dynamic LSI-based semantic index generation for MP2P semantic information retrieval systems is achieved.

Two measurements, coverage speed and update cost, are considered to evaluate the proposed CSU. By the analysis, the average time for an update request to update the whole network grows in the order of $O(\sqrt[d]{n})$ when the underlying MP2P network is a d-dimensional CAN. For k requests, each request costs at most log k times the number of messages for a request in contention-free environments in average. When the contention-avoid strategy is applied, the update cost per update can be expected to be linear proportional to the number of simultaneously issued requests due to the avoidance of conflicts. To validate the proposed CSU, we perform the simulated experiments extensively. The experimental results show the same trends as the analytical results and demonstrate the effectiveness of CSU on updating the concept space with LSI/SVD index structure on MP2P semantic IR systems. In the future, we would like to explore further about the impact of the dynamic features in MP2P systems on the performance, including the node mobility models, node speed, network distribution, network density and the arrival models of update messages.

References

[1] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proc. of the First IEEE Int. Conf. on Peer-to-Peer Computing*, pp. 101, 2001. Article (CrossRef Link)

[2] C.-C. Lai, C.-M. Liu and Y.-C. Su, "A novel mechanism to construct a compatible overlay on heterogeneous mobile peers," in *Proc. of the 2013 IEEE Int. Conf. on Pervasive Computing and Communications Workshops*, pp.78-83, 2013. Article (CrossRef Link)

[3] D. Zeinalipour-Yazti, V. Kalogeraki and D. Gunopulos, "Exploiting locality for scalable information retrieval in peer-to-peer networks," *Information Systems*, vol. 30, no. 4, pp.277-298, 2005. Article (CrossRef Link)

[4] C. Tang, Z. Xu and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *Proc. of the ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 175-186, 2003. Article (CrossRef Link)

[5] Y. Chen, Z. Xu and C. Zhai, "A scalable semantic indexing framework for peer-to-peer information retrieval," in *Proc. of the ACM SIGIR Workshop on Heterogeneous and Distributed Information Retrieval*, 2005. Article (CrossRef Link)

[6] R. A Ferreira, M. Koyut¨urk, S. Jagannathan and A. Grama, "Semantic indexing in structured peer-to-peer networks," *Journal of Parallel and Distributed Computing*, vol.68, no. 1, pp.64-77, 2008. Article (CrossRef Link)

[7] T. G. Kolda and D. P. O'Leary, "A semidiscrete matrix decomposition for latent semantic indexing information retrieval," *ACM Transactions on Information Systems*, vol.16, no.4, pp.322-346, 1998. Article (CrossRef Link)

[8] S. Zhang, G. Wu, G. Chen, and L. Xu, "On building and updating distributed LSI for P2P systems," in *Proc. of Parallel and Distributed Processing and Applications - ISPA 2005 Workshops*, pp. 9-16, 2005. Article (CrossRef Link)
[9] H. T. Shen, Y. Shu, and B. Yu, "Efficient semantic-based content search in P2P network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 813-826, 2004. Article (CrossRef Link)
[10] I. Rudomilov and I. Jelinek, "Semantic p2p search engine," in *Proc. of the Federated Conf. on Computer Science and Information Systems*, pp. 991-995, 2011. Article (CrossRef Link)
[11] Y. Yang, "Semantic Information Retrieval over P2P Network," *Les 6è Rencontres Jeunes Chercheurs en Recherche d'Information associé Conférence en Recherche d'Information et Applications*, pp. 391-396, 2011. Article (CrossRef Link)
[12] Y. Yang, "Semi-structured semantic overlay for information retrieval in self-organizing networks," in *Proc. of the 21st Int. Conf. Companion on World Wide Web*, pp. 203-208, 2012. Article (CrossRef Link)
[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391-407, 1990. Article (CrossRef Link)
[14] A. Kontostathis, "Essential dimensions of latent semantic indexing (LSI)," in *Proc. of the IEEE Hawaii Int. Conf. on System Sciences*, pp. 73, 2007. Article (CrossRef Link)
[15] E. Vecharynski and Y. Saad, "Fast updating algorithms for latent semantic indexing," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no.6, pp. 1105-1131, 2014. Article (CrossRef Link)
[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Schenker, "A scalable content-addressable network," in *Proc. of the ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 161-172, 2001. Article (CrossRef Link)
[17] I. Hegedus, M. Jelasity, L. Kocsis, and A. A. Benczur, "Fully distributed robust singular value decomposition," in *Proc. of* the 14th IEEE International Conference on Peer-to-Peer Computing, pp.1-9, 2014. Article (CrossRef Link)
[18] G. Weikum and G. Vossen, *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. Morgan Kaufmann Publishers Inc., 2001. Article (CrossRef Link)
[19] H. Zha and H. D. Simon, "On updating problems in latent semantic indexing," *SIAM Journal on Scientific Computing*, vol. 21, no. 2, pp.782-791, 1999. Article (CrossRef Link)
[20] 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 Transactions on Networking*, vol. 11, no.1, pp. 17-32, 2003. Article (CrossRef Link)
[21] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms Heidelberg*, pp. 329-350, 2001. Article (CrossRef Link)
[22] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," I*EEE Journal on Selected Areas in Communications*, vol. 22, pp. 41-53, 2004. Article (CrossRef Link)
[23] P. Maymounkov and D. Mazi`eres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. of the 1st Int. Workshop on Peer-to-Peer Systems*, pp. 53-65, 2002. Article (CrossRef Link)
[24] M. Berry, T. Do, G. O'Brien, V. Krishna and S. Varadhan, "Svdpackc user's guide," *Technical report*, Department of Computer Science, University of Tennessee, 1993. Article (CrossRef Link)
[25] M. W. Berry, S. T. Dumais and T. A. Letsche, "Computational methods for intelligent information access," in *Proc. of the ACM Conf. on Supercomputing*, pp. 20, 1995. Article (CrossRef Link)
[26] V. Mesaros, R. Collet, K. Glynn and P. Van Roy, "A transactional system for structured overlay networks," *Technical Report*, Louvain-la-Neuve, BLN, FR, 2005. Article (CrossRef Link)
[27] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *ACM Communication*, vol. 21, pp. 1040-1048, 1978. Article (CrossRef Link)

Chuan-Ming Liu is an associate professor in the Department of Computer Science and Information Engineering, National Taipei University of Technology (NTUT), TAIWAN. He received his Ph. D. in Computer Sciences from Purdue University in 2002 and B.S. and M.S. degrees both in Applied Mathematics from National Chung-Hsing University, Taiwan, in 1992 and 1994, respectively. In the summer of 2010 and 2011, he has held visiting appointments at Auburn University and Beijing Institute of Technology, respectively. Dr. Liu's research interests include data management and data dissemination in various emerging computing environments, query processing in moving objects, location-based services, ad-hoc and sensor networks, parallel and distributed computation, and analysis and design of algorithms

Cheng-Hsien Chen is a Senior S/W Developer in Foxconn Internal Holdings, New Taipei City, TAIWAN. He received his B.S. and M.S. degree in Computer Sciences of Information Engineering both from National Taipei University of Technology, Taiwan, in 2008 and 2010, respectively. He is currently working on sophisticated modern mobile systems and pragmatic experienced with parts of S/W constructions & architecture with particular mobile products. His greatest expertise revolves in the areas of concurrent S/W design, system performance tuning, and multimedia solutions. He has strong knowledge about how the computations should be constructed to adapt a productive mobile device.

Yen-Lin Chen received a Ph.D. in Electrical and Control Engineering from National Chiao Tung University (Hsinchu, Taiwan) in 2006. From August 2009 to January 2012, he worked as an Assistant Professor in the Department of Computer Science and Information Engineering at National Taipei University of Technology (Taipei, Taiwan), where he has worked as an Associate Professor since February, 2012. His research interests include image and video processing, embedded systems, pattern recognition, intelligent vehicles, and intelligent human-computer interactive systems.

Jenq-Haur Wang is an associate professor in Department of Computer Science and Information Engineering, National Taipei University of Technology, Taiwan. He received his B.S. and Ph.D degrees in Computer Science from National Taiwan University, Taiwan, in 1994 and 2002, respectively. In 2014, he was invited as a visiting professor at Singapore Management University. His research interests include social Web mining, big data analytics, information security, distributed computing, and peer-to-peer retrieval.