IEEE Access
The journal for rapid open access publishing

Received March 24, 2015, accepted May 26, 2015, date of publication June 12, 2015, date of current version July 3, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2444872

Improving the Search Mechanism for Unstructured Peer-to-Peer Networks Using the Statistical Matrix Form

CHIA-HUNG LIN, JING-JIA ZSENG, AND SUN-YUAN HSIEH
Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan
Corresponding author: S.-Y. Hsieh (hsiehsy@mail.ncku.edu.tw)

A preliminary version of this paper appeared in [Chia-Hung Lin and Sun-Yuan Hsieh "A New Search Mechanism for Unstructured Peer-to-Peer Networks" IEEE.AM Conference, Proceedings of the 2nd International Conference on Applied Informatics and Computing Theory (AICT '11), pp. 195-200, 2011].

ABSTRACT In a traditional file search mechanism, such as flooding, a peer broadcasts a query to its neighbors through an unstructured peer-to-peer (P2P) network until the time-to-live decreases to zero. A major disadvantage of flooding is that, in a large-scale network, this blind-choice strategy usually incurs an enormous traffic overhead. In this paper, we propose a method, called the statistical matrix form (SMF), which improves the flooding mechanism by selecting neighbors according to their capabilities. The SMF measures the following peer characteristics: 1) the number of shared files; 2) the content quality; 3) the query service; and 4) the transmission distance between neighbors. Based on these measurements, appropriate peers can be selected, thereby reducing the traffic overhead significantly. Our experimental results demonstrate that the SMF is effective and efficient. For example, compared with the flooding search mechanism in dynamic unstructured P2P networks, the SMF reduces the traffic overhead by more than 80%. Moreover, it achieves a good success rate and shorter response times.

INDEX TERMS Unstructured peer-to-peer networks, flooding search mechanism, traffic overhead, statistical matrix form.

I. INTRODUCTION

Generally, Peer-to-Peer (P2P) networks can be classified as: *structured P2P networks*, which are based on centralized management (e.g., Chord [2], Pastry [8]), and *unstructured P2P networks*, which are built on a distributed search mechanism (e.g., Gnutella [5], FrostWire [4]). Although both types allow users to participate in a fully distributed cooperative network, unstructured P2P networks give participants more freedom to exchange resources and services. The major disadvantage of unstructured P2P networks is that their basic search mechanism, "flooding," incurs an enormous traffic overhead. To resolve this issue, numerous search mechanisms have been proposed to replace or improve the flooding mechanism [10], [15], [21], [36], [46], [58]. In this paper, our object is to improve the flooding mechanism by exploiting the scalability of unstructured P2P networks.

In a real network environment, peers differ from each other in a number of respects, such as the number of shared files, the content quality, the query service and transmission distance between neighbors. These characteristics are crucial because they can be utilized to optimize the search performance effectively. We propose a method that statistically analyzes query messages in terms of the following four characteristics: *Processing Ability* (*PA*), *Effective Sharing* (*ES*), *Index Power* (*IP*), and *Transmission Efficiency* (*TE*). The *PA* of peers is analyzed to determine which peers leech the most resources without giving feedback [9], [29], [35], [68]. The *ES* refers to the number of files that a peer shares, and can be used to classify a peer's sharing capability. It has been shown that, in a network, very few peers share a large number files, so that the quality of the files influences the sharing capability [54], [65], [70], [74]. The *IP* measures the number of files that a peer records in the index cache, and can also be used to analyze the number of responses in the cache content. Finally, the *TE* is utilized to measure the distance between peers in order to prevent inefficient routing. Xiao *et al.* [67] observe that only a small number of links connect peers in the same Autonomous System (AS).[1] Most generated connections stray past AS borders to produce more distant links [14], [49].

[1] An autonomous system is sometimes referred to as a routing domain, which is a collection of IP networks and routers under the control of one entity.

IEEE Access
The journal for rapid open access publishing

We represent the four characteristics in a matrix form called the Statistical Matrix Form (SMF). To adjust the values of the matrix, we utilize a standard deviation technique to determine an overall ranking of a query peer's neighbors. As a result, the performance of the flooding search mechanism can be improved by only sending query messages to the top-k ranked neighbors of a query peer for some k determined by careful analysis, instead of sending messages to all the peer's neighbors. The performance evaluation demonstrates that the response time and traffic overhead can be reduced significantly, while the computation overhead is acceptable.

The remainder of this paper is organized as follows. Section II contains a review of related work. In Section III, we introduce the proposed SMF search mechanism; and in Section IV, we evaluate the mechanism's performance. Then, in Section V, we summarize our conclusions.

II. RELATED WORK

Numerous search mechanisms have been proposed to reduce the large amount of unnecessary traffic generated by flooding-based search mechanisms in unstructured P2P networks. The flooding technique sends query messages to all the logical neighbors of a query peer, except the incoming peer, until the Time-To-Live[2] (TTL) decreases to zero or the query receives a response. It has been shown that the *Random Walk* (RW) approach reduces the exponentially increasing flood traffic caused by randomly choosing a neighbor to send a query message until sufficient responses are generated [22], [46]. Although the amount of traffic can be reduced, the RW search mechanism suffers from two fundamental problems. First, it is essentially a blind search because, in each step, a query is forwarded to a random peer. Second, if the query arrives at a peer that is already overloaded with traffic, the queried peer may be queued for an excessive amount of time before it can be handled. The random *k-walker* algorithm [69] improves RW by sending a query to k neighbors, called "k-walkers," of the source peer. Each walker randomly selects one neighbor and delivers the query to that neighbor. The walkers' queries are processed sequentially. The *Multiple Random Walk* (MRW) [19], [36], [66] improves the RW and the k-walker approaches by enabling a query peer to select k different neighbors arbitrarily in each step. The larger the number of peers that the query is delivered to, the more opportunities the query peer will have to find hits. As the search mechanism increases the search scope, the query success rate can be improved. The above search mechanisms select neighbors without any strategies, so their performance may not be satisfactory. To address the problem, three other types of search mechanisms have been proposed.

The first type utilizes an index-based strategy, which records the query strings and outcomes that flow through queried peers from each response peer. Under this strategy, a peer receives a query from its neighbor and forwards the message to the mapping peer directly based on its routing table (index). For structured P2P networks, the *Distributed Hash Table* (DHT) technique [20], [52], [53], [57], [71] has been proposed to improve the search performance of index-based search mechanisms. However, under this technique, it is difficult to control the traffic overhead and maintain indices correctly in highly dynamic P2P networks. Meanwhile, for unstructured P2P networks, the *Uniform Index Caching* (UIC) technique [48] generates a large number of redundant records in each index [55]. As a result, improvements of UIC are proposed in [25], [26], [59], and [64].

The second type of search mechanism attempts to construct an optimized overlay topology. In unstructured P2P systems, peers join and leave the network in a random fashion, which induces the so-called *topology mismatch problem* [23], [31], [45] between the physical and logical networks. To reduce the traffic overhead in an inefficient topology, several algorithms have been developed to transform the original topology into a minimum spanning tree (MST) [33], [43], [67]. Alternatively, analysis of the index content or user query history can be used to construct a tree-based logical topology under some mathematical model [11], [30]–[32], [34].

The third type of search mechanism is called the *Adaptive Probabilistic Search* (APS) approach [63] in which a query peer only sends a query to a proper subset of appropriate neighbors rather than all of its neighbors. In this mechanism, choosing a subset of influential neighbors without a high overhead is the most important factor [16], [21], [37], [62]. In [11], [47], and [50], each peer records the results received from its neighbors and each link's transmission time in its routing table. The algorithms proposed in [25], [55], [59], [64], and [72] construct filters by designing hash functions to compute the capability of a query peer's neighbors. In addition, Chen *et al.* [15] exploit users' common interest patterns captured by a probability-theoretic framework to select a subset of neighbors. The forwarding-based algorithm is another method used to improve the search performance [21], [27], [40], [41]. Gkantsidis *et al.* [22] determine the number of neighbors in query process based on the value of TTL. By utilizing the search mechanism, the traffic cost can be reduced and the same flooding scope can be maintained [40], [62].

Among the above search mechanisms, APS is the most influential and most suitable approach for unstructured P2P networks. However, like the other approaches, APS only considers a few features in the flooding-based query process. This observation motivated us to design the SMF, which includes more useful features, to optimize the search performance.

III. THE SEARCH MECHANISM BASED ON THE SMF

The SMF of query peer u is comprised of two matrixes: the *left-hand matrix* and the *right-hand matrix*.

[2]A value in an Internet Protocol (IP) packet that tells a network router whether or not the packet has been in the network too long and should be discarded.

The *left-hand matrix*, called the *feature matrix* (*FM*), is an $n \times 4$ matrix, where n is the number of neighbors of u. To derive the entries for the four columns of the *FM*, we compute the *PA*, *ES*, *IP*, and *TE* scores for n neighbors of u by the method described in Subsections III-A–III-E; then, we record the computed scores in the first, second, third, and fourth columns respectively. The right-hand matrix, called the *weight matrix* (*WM*), is a 4×1 matrix in which each peer can set the proper weights according to the derivation degree of each feature. The method is described in Section III-F. Finally, each query peer u computes a *scoring matrix* (*SM*), which is an $n \times 1$ matrix obtained by the matrix multiplication $FM \times WM$. To deliver queries for u, we obtain the score of each of u's neighbors in the *SM* and then select the neighbors with the top-k scores to send query messages. Figure 1 illustrates the structure of SMF. Since the query peer u has five neighbors, v, w, x, y, and z, its feature matrix is a 5×4 matrix; the weight matrix is a 4×1 matrix; and the score matrix is computed by the formula $SM = FM \times WM$, which is a 5×1 matrix.

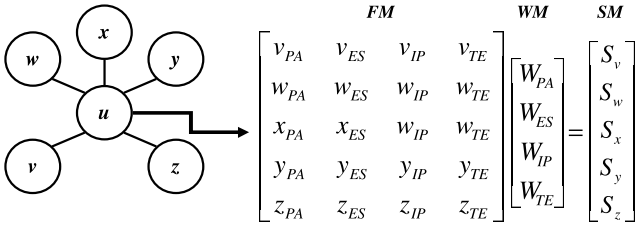


FIGURE 1. An example of the SMF for a query peer *u*.

A. FEATURE COLLECTION SCOPE

We define the *d-collected scope* of a query peer u as the set of peers that are at most d hop(s) away from u. In the construction of the *FM* presented in the following subsections, we collect relative information about the d-collected scope of a query peer. For example, the information about the 2-collected scope of a query peer u contains the information about each neighbor v of u, as well as that of v's neighbors. We could improve the search performance by increasing the value of d; however, it may increase the computational overhead because of the extra cost of collecting information. Therefore, the problem is how to choose appropriate values of d to improve the performance by determining the acceptable extra-overhead incurred by the collection and exchange of information. The experimental results reported in Section IV-D2 demonstrate that $d = 2$ or $d = 3$ are the optimal values; hence, we adopt $d = 2$ in the remainder of this paper.

B. PROCESSING ABILITY (PA)

In P2P networks, there are usually free loaders[3] [9], [35], [68] who download files without sharing any of their resources, which impacts the search performance of coadjutant communities. To prevent free loaders, we utilize the *PA* to differentiate between leeching and enthusiastic peers. The *PA* score is computed in terms of the peers' query frequency and response frequency, which we discuss in the next two subsections.

1) QUERY FREQUENCY (QF)

In a P2P network, a query peer that generates a lot of queries may be a free loader. Let $N(u)$ be the neighbors of a query peer u; that is, $N(u)$ are peers that are one hop away from u. In addition, let $NQ(v)$ be the number of queries sent by v.

Each query peer u computes $SQ_1(u)$, which is the total number of queries (SQ) sent from the peers that are one hop away from u. Formally,

$$SQ_1(u) = \sum_{v \in N(u)} NQ(v). \tag{1}$$

The *Query-Minus-Score (QMS) of a neighbor v of u* is defined as

$$QMS(u, v) = SQ_1(u) - NQ(v). \tag{2}$$

When $NQ(v)$ increases, the possibility of v being regarded as a free loader also increases and peer v will be assigned a lower score.

Next, each query peer u computes $SQMS_1(u)$ (resp. $SQMS_2(u)$), which is the sum of the query-minus-scores (*SQMS*) of all peers that are one (resp. two) hop(s) away from u:

$$SQMS_1(u) = \sum_{v \in N(u)} QMS(u, v) \tag{3}$$

and

$$SQMS_2(u) = \sum_{v \in N(u)} SQMS_1(v). \tag{4}$$

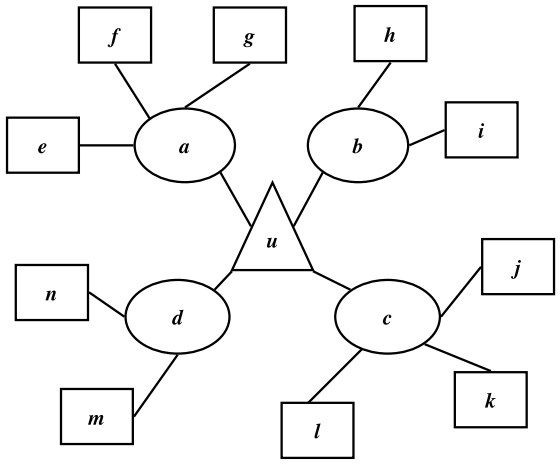
Then, the *query frequency of a neighbor v of u* is defined as

$$QF(u, v) = w_1 * \frac{QMS(u, v)}{SQMS_1(u)} + w_2 * \frac{SQMS_1(v)}{SQMS_2(u)}, \tag{5}$$

where w_1 and w_2 are two parameters used to adjust the influence of peers that are one hop away and two hops away from u respectively. Based on Eq. (5), a peer can determine the amount of resources that their neighbors leech from the network.

Example 1: Consider the P2P network shown in Figure 2. Based on Eq. (1), $SQ_1(u) = NQ(a) + NQ(b) + NQ(c) + NQ(d) = 3 + 6 + 5 + 4 = 18$; $SQ_1(a) = NQ(e) + NQ(f) + NQ(g) = 4 + 2 + 5 = 11$; $SQ_1(b) = NQ(h) + NQ(i) = 3 + 1 = 4$; $SQ_1(c) = NQ(j) + NQ(k) + NQ(l) = 3 + 7 + 2 = 12$; and $SQ_1(d) = NQ(m) + NQ(n) = 1 + 6 = 7$. Next, we compute the query-minus-score of each peer as $QMS(u, a) = SQ_1(u) - NQ(a) = 18 - 3 = 15$, $QMS(u, b) = SQ_1(u) - NQ(b) = 18 - 6 = 12$, $QMS(u, c) = SQ_1(u) - NQ(c) = 18 - 5 = 13$, and $QMS(u, d) = SQ_1(u) - NQ(d) = 18 - 4 = 14$.

[3]Peers who use resources without contributing anything reduce a network's overall service capacity.



(a)

Feature / Peer	*PA*		*ES*		*IP*		*TE*
	NQ	*NR*	*NF*	*NFH*	*NI*	*NIH*	*LD* (ms)
a	3	21	30	18	20	12	103
b	6	12	38	22	28	16	208
c	5	16	27	16	17	10	287
d	4	33	42	25	32	19	50
e	4	34	49	29	39	23	141
f	2	22	43	25	33	19	112
g	5	5	22	13	12	7	58
h	3	37	25	15	15	9	211
i	1	3	21	12	11	6	151
j	3	41	28	16	18	10	78
k	7	22	40	24	30	18	22
l	2	33	58	34	48	28	106
m	1	11	45	27	35	21	77
n	6	26	32	19	22	13	28

(b)

FIGURE 2. Illustration of Example 1: (a) A P2P network. (b) The values of the four features of the peers that are at most two hops away from *u*.

Then, based on Eq. (3), $SQMS_1(u) = QMS(u,a) + QMS(u,b) + QMS(u,c) + QMS(u,d) = 15 + 12 + 13 + 14 = 54$; and based on Eq. (4), $SQMS_2(u) = SQMS_1(a) + SQMS_1(b) + SQMS_1(c) + SQMS_1(d) = (QMS(a,e) + QMS(a,f) + QMS(a,g)) + (QMS(b,h) + QMS(b,i)) + (QMS(c,j) + QMS(c,k) + QMS(c,l)) + (QMS(d,m)+QMS(d,n)) = ((SQ_1(a)-NQ(e))+(SQ_1(a) - NQ(f)) + (SQ_1(a) - NQ(g))) + ((SQ_1(b) - NQ(h)) + (SQ_1(b)-NQ(i)))+((SQ_1(c)-NQ(j))+(SQ_1(c)-NQ(k)) + (SQ_1(c) - NQ(l))) + ((SQ_1(d) - NQ(m)) + (SQ_1(d) - NQ(n))) = ((11-4)+(11-2)+(11-5))+((4-3)+(4-1))+((12-3)+(12-7)+(12-2))+((7-1)+(7-6)) = (7+9+6)+(1+3)+(9+5+10)+(6+1) = 57$. If we set $w_1 = 1$ and $w_2 = 2$, then, according to Eq. (5),

$$QF(u,a) = 1 * \frac{15}{54} + 2 * \frac{7+9+6}{57} = 1.0498,$$
$$QF(u,b) = 1 * \frac{12}{54} + 2 * \frac{1+3}{57} = 0.3626,$$
$$QF(u,c) = 1 * \frac{13}{54} + 2 * \frac{9+5+10}{57} = 1.0829, \text{ and}$$
$$QF(u,d) = 1 * \frac{14}{54} + 2 * \frac{6+1}{57} = 0.5049.$$

2) RESPONSE FREQUENCY (RF)

If a peer responds to a large number of queries, we define it as an "eager" peer. The term "response frequency" refers to a peer's ability to respond to queries.

Each peer u computes $SR_1(u)$ (resp. $SR_2(u)$), which is the sum of the response times (SR) of peers that are one (resp. two) hop(s) away from u. Formally,

$$SR_1(u) = \sum_{v \epsilon N(u)} NR(v), \tag{6}$$

where $NR(v)$ is the number of responses sent by peer v, and

$$SR_2(u) = \sum_{v \epsilon N(u)} SR_1(v). \tag{7}$$

Based on the above two equations, the *response frequency of a neighbor v of u*, denoted by $RF(u,v)$, is computed as follows:

$$RF(u,v) = w_1 * \frac{NR(v)}{SR_1(u)} + w_2 * \frac{SR_1(v)}{SR_2(u)}. \tag{8}$$

From $RF(u,v)$, we can determine the response ability of a neighbor v.

Example 2: Let us consider Figure 2 again. First, peer u computes $SR_1(u) = NR(a) + NR(b) + NR(c) + NR(d) = 21 + 12 + 16 + 33 = 82$ (by Eq. (6)), and $SR_2(u) = SR_1(a) + SR_1(b) + SR_1(c) + SR_1(d) = (NR(e) + NR(f) + NR(g)) + (NR(h) + NR(i)) + (NR(j) + NR(k) + NR(l)) + (NR(m) + NR(n)) = (34 + 22 + 5) + (37 + 3) + (41 + 22 + 33) + (11 + 26) = 234$ (by Eq. (7)) to realize the response status of each of its neighbors. Then, based on Eq. (8), we can compute the response frequency of each neighbor of u as follows:

$$RF(u,a) = 1 * \frac{21}{82} + 2 * \frac{34+22+5}{234} = 0.7775,$$
$$RF(u,b) = 1 * \frac{12}{82} + 2 * \frac{37+3}{234} = 0.4882,$$
$$RF(u,c) = 1 * \frac{16}{82} + 2 * \frac{41+22+33}{234} = 1.0156, \text{ and}$$
$$RF(u,d) = 1 * \frac{33}{82} + 2 * \frac{11+26}{234} = 0.7186.$$

Since the response frequency of peer c is much greater than that of peers a, b, and d, we regard c as an eager peer.

We represent the *processing ability of a neighbor v of u*, denoted by $PA(u,v)$, in terms of the query frequency and the response frequency as

$$PA(u,v) = QF(u,v) + RF(u,v). \tag{9}$$

Example 3: Based on the values derived in Example 1 and Example 2, we can compute the processing ability of each neighbor of peer u in Figure 2 as follows:

$$\begin{aligned} PA(u,a) &= QF(u,a) + RF(u,a) \\ &= 1.0498 + 0.7775 = 1.8273, \\ PA(u,b) &= QF(u,b) + RF(u,b) \\ &= 0.3626 + 0.4882 = 0.8508, \\ PA(u,c) &= QF(u,c) + RF(u,c) \\ &= 1.0829 + 1.0156 = 2.0985, \text{ and} \\ PA(u,d) &= QF(u,d) + RF(u,d) \\ &= 0.5049 + 0.7186 = 1.2595. \end{aligned}$$

C. EFFECTIVE SHARING (ES)

This feature is based on the observation in [38], [54], [65], and [74] that the file-sharing among peers is extremely unbalanced. For example, it has been shown that 7 percent of peers in a P2P network share more files than all the other peers can provide, and the top-1 percent of peers respond to 47 percent of the queries. Instead of all peers participating in file-sharing, only a small number of volunteer peers provide most of the resource sharing services in a P2P network. Moreover, peers' query response capabilities vary because of the heterogeneity of their file-sharing resources. The trace analysis in [59] showed that a small number of peers share a large number of files. Because query answering involves matching keywords with the names of all shared files, we posit that, as the number of shared files increases, the probability of successful matching should also increase. Based on the above observations, we propose the concept of *effective sharing* (ES), which is used to determine the number of files shared among peers in a P2P network. The ES is comprised of two sub-features: the *sharing count* (*SC*), described in the next subsection III-C1; and the *quality of sharing* (*QS*), described in Subsection III-C2.

1) SHARING COUNT (SC)

When choosing influential neighbors to send queries from a query peer, it is necessary to consider the number of files shared by the peers. In a real environment, if a peer shares a large number of files, it should have a higher probability of matching queries than a peer that only shares a few files. Each query peer u computes $SF_1(u)$ (resp. $SF_2(u)$) which is the total number of shared files (SF) by peers that are one (resp. two) hop(s) away from u. Formally,

$$SF_1(u) = \sum_{v \in N(u)} NF(v), \tag{10}$$

where $NF(v)$ is the number of shared files, and

$$SF_2(u) = \sum_{v \in N(u)} SF_1(v). \tag{11}$$

The *sharing count of a neighbor v of u* is defined as

$$SC(u,v) = w_1 * \frac{NF(v)}{SF_1(u)} + w_2 * \frac{SF_1(v)}{SF_2(u)}. \tag{12}$$

Based on Eq. (12), if the SC score of some neighbor v of a query peer u is the largest among all the neighbors, then it is regarded as the most influential neighbor. The reason is that v shares more files than the other neighbors, so there is a much higher probability that u will get responses from v.

Example 4: In Figure 2, to determine the file sharing status of its neighbors, peer u computes $SF_1(u) = NF(a)+NF(b)+NF(c)+NF(d) = 30+38+27+42 = 137$ by Eq. (10), and $SF_2(u) = SF_1(a)+SF_1(b)+SF_1(c)+SF_1(d) = (NF(e)+NF(f)+NF(g))+(NF(h)+NF(i))+(NF(j)+NF(k)+NF(l))+(NF(m)+NF(n)) = (49+43+22)+(25+21)+(28+40+58)+(45+32) = 363$ by Eq. (11). Moreover, according to Eq. (12), we have

$$\begin{aligned} SC(u,a) &= 1 * \frac{30}{137} + 2 * \frac{49+43+22}{363} = 0.8471, \\ SC(u,b) &= 1 * \frac{38}{137} + 2 * \frac{25+21}{363} = 0.5308, \\ SC(u,c) &= 1 * \frac{27}{137} + 2 * \frac{28+40+58}{363} = 0.8913, \text{ and} \\ SC(u,d) &= 1 * \frac{42}{137} + 2 * \frac{45+32}{363} = 0.7308. \end{aligned}$$

Since peer c has the highest value, it is the most influential neighbor peer for file sharing.

2) QUALITY OF SHARING (QS)

It has been shown that some shared files are never used to answer queries [9], [35]. If we only consider the number of files used to answer queries, the number of files shared with query peers has a strong correlation with the responding peers. Motivated by this fact, we utilize the second *ES* sub-feature, the "quality of sharing", to distinguish useful files from useless files. Let $NFH(v)$ be the number of v's shared files that match queries. Each query peer u computes $SFH_1(u)$ (resp. $SFH_2(u)$), which is the effectiveness of the neighbors of a query peer u, by summing the NFH values of the peers that are one (resp. two) hop(s) away from u. Formally,

$$SFH_1(u) = \sum_{v \in N(u)} NFH(v) \tag{13}$$

and

$$SFH_2(u) = \sum_{v \in N(u)} SFH_1(v). \tag{14}$$

The *quality of sharing* (*QS*) *of a neighbor v of u* is defined as

$$QS(u,v) = w_1 * \frac{NFH(v)}{SFH_1(u)} + w_2 * \frac{SFH_1(v)}{SFH_2(u)}. \tag{15}$$

Based on Eq. (15), if the QS score of some neighbor v of a query peer u is the largest among all the neighbors, then v is regarded as the most influential neighbor because it has the most opportunities to share files with the query peer u.

Example 5: In Figure 2, peer u computes $SFH_1(u) = NFH(a)+NFH(b)+NFH(c)+NFH(d) = 18+22+16+25 = 81$ by Eq. (13), and

$SFH_2(u) = SFH_1(a) + SFH_1(b) + SFH_1(c) + SFH_1(d) = (NFH(e) + NFH(f) + NFH(g)) + (NFH(h) + NFH(i)) + (NFH(j) + NFH(k) + NFH(l)) + (NFH(m) + NFH(n)) = (29+25+13)+(15+12)+(16+24+34)+(27+19) = 214$ by Eq. (14). According to Eq. (15), we have

$$QS(u,a) = 1 * \frac{18}{81} + 2 * \frac{29+25+13}{214} = 0.8484,$$
$$QS(u,b) = 1 * \frac{22}{81} + 2 * \frac{15+12}{214} = 0.5239,$$
$$QS(u,c) = 1 * \frac{16}{81} + 2 * \frac{16+24+34}{214} = 0.8891, \text{ and}$$
$$QS(u,d) = 1 * \frac{25}{81} + 2 * \frac{27+19}{214} = 0.7385.$$

We then represent the *effective sharing of a neighbor v of u*, denoted by $ES(u, v)$, in terms of the sharing count and the quality of sharing as follows:

$$ES(u,v) = SC(u,v) + QS(u,v). \tag{16}$$

Example 6: Based on Eq. (16) and the results derived in Examples 4 and 5, we have

$$\begin{aligned} ES(u,a) &= SC(u,a) + QS(u,a) \\ &= 0.8471 + 0.8484 = 1.6955, \\ ES(u,b) &= SC(u,b) + QS(u,b) \\ &= 0.5308 + 0.5239 = 1.0547, \\ ES(u,c) &= SC(u,c) + QS(u,c) \\ &= 0.8913 + 0.8891 = 1.7804, \text{ and} \\ ES(u,d) &= SC(u,d) + QS(u,d) \\ &= 0.7308 + 0.7385 = 1.4693. \end{aligned}$$

D. INDEX POWER (IP)

In a P2P network, volunteers have different-sized indexes to record historical information. However, if a peer records a large number of file sharing messages in its index, many of the messages may never be used by the mechanism. The *Index Power* (*IP*) feature determines the amount of content in a queried peer's index and assesses its quality. The IP comprises two sub-features: *Index Counting* (*IC*) described in the next subsection and *Quality of the Index* (*QI*) described in Subsection III-D2.

1) INDEX COUNT (IC)

The index count feature records the number of messages in a peer's index. We assume that if a peer records a large amount of information in its index, it will have a higher probability of matching queries. Each query peer u computes $SI_1(u)$ (resp. $SI_2(u)$) which is the number of indices of the peers that are one (resp. two) hop(s) away from u. Formally,

$$SI_1(u) = \sum_{v \in N(u)} NI(v), \tag{17}$$

where $NI(v)$ is the number of index records in peer v, and

$$SI_2(u) = \sum_{v \in N(u)} SI_1(v). \tag{18}$$

Then, the *index count of a neighbor v of u* is calculated as follows:

$$IC(u,v) = w_1 * \frac{NI(v)}{SI_1(u)} + w_2 * \frac{SI_1(v)}{SI_2(u)}. \tag{19}$$

Based on Eq. (19), if the *IC* score of some neighbor v of a query peer u is the largest among all the other neighbors, then v can be regarded as the most influential neighbor because it has the largest possibility to reply the query sent from the query peer u via v's index.

Example 7: Returning to Figure 2, to determine its neighbors' index sharing status, peer u first computes $SI_1(u) = NI(a) + NI(b) + NI(c) + NI(d) = 20 + 28 + 17 + 32 = 97$ by Eq. (17), and then computes $SI_2(u) = SI_1(a)+SI_1(b)+SI_1(c)+SI_1(d) = (NI(e)+NI(f)+NI(g))+(NI(h)+NI(i))+(NI(j)+NI(k)+NI(l))+(NI(m)+NI(n)) = (39+33+12)+(15+11)+(18+30+48)+(35+22) = 263$ by Eq. (18). Then, based on Eq. (19), we have

$$IC(u,a) = 1 * \frac{20}{97} + 2 * \frac{39+33+12}{263} = 0.8450,$$
$$IC(u,b) = 1 * \frac{28}{97} + 2 * \frac{15+11}{263} = 0.4864,$$
$$IC(u,c) = 1 * \frac{17}{97} + 2 * \frac{18+30+48}{263} = 0.9053, \text{ and}$$
$$IC(u,d) = 1 * \frac{32}{97} + 2 * \frac{35+22}{263} = 0.7634.$$

Since peer c has the highest value, it has the largest possibility to reply the query sent from the query peer u via c's index.

2) QUALITY OF THE INDEX (QI)

The second sub-feature analyzes the quality of an index's content and the characteristics of the files. Since the probability of index hits may be influenced by the index counts as well as the quality of the index's content, we count the number of index hits to analyze the quality of the information in the index. Each query peer u computes $SIH_1(u)$ (resp. $SIH_2(u)$) which is the total number of index hits of peers that are one (resp. two) hop(s) away from u. Formally,

$$SIH_1(u) = \sum_{v \in N(u)} NIH(v), \tag{20}$$

where $NIH(v)$ is the number of index hits of a neighbor v of u, and

$$SIH_2(u) = \sum_{v \in N(u)} SIH_1(v). \tag{21}$$

Based on Eq. (20) and Eq. (21), $QI(u, v)$ can be computed as follows:

$$QI(u,v) = w_1 * \frac{NIH(v)}{SIH_1(u)} + w_2 * \frac{SIH_1(v)}{SIH_2(u)}. \tag{22}$$

Based on Eq. (22), if the *QI* score of some neighbor v of a query peer u is the largest score, then v is regarded as the most influential neighbor for index hits, i.e., it has the highest probability of responding to the query sent by u.

Example 8: In Figure 2, to obtain the number of its neighbors' index hits, peer u first computes $SIH_1(u) = NIH(a) + NIH(b) + NIH(c) + NIH(d) = 12 + 16 + 10 + 19 = 57$ by Eq. (20), and then computes $SIH_2(u) = SIH_1(a) + SIH_1(b) + SIH_1(c) + SIH_1(d) = (NIH(e)+NIH(f)+NIH(g))+(NIH(h)+NIH(i))+(NIH(j)+NIH(k)+NIH(l))+(NIH(m)+NIH(n)) = (23+19+7)+(9+6)+(10+18+28)+(21+13) = 154$ by Eq. (21). Then, according to Eq. (22), we have

$$QI(u,a) = 1 * \frac{12}{57} + 2 * \frac{23+19+7}{154} = 0.8469,$$
$$QI(u,b) = 1 * \frac{16}{57} + 2 * \frac{9+6}{154} = 0.4755,$$
$$QI(u,c) = 1 * \frac{10}{57} + 2 * \frac{10+18+28}{154} = 0.9027, \text{ and}$$
$$QI(u,d) = 1 * \frac{19}{57} + 2 * \frac{21+13}{154} = 0.7749.$$

Since peer c has the highest value, it has the highest probability of responding to the query sent by u.

We then represent the *index power of a neighbor v of u*, denoted by $IP(u, v)$, in terms of the index count and the quality of the index as follows:

$$IP(u,v) = IC(u,v) + QI(u,v). \tag{23}$$

Example 9: According to Eq. (23) and the results derived in Examples 7 and 8, we have

$$\begin{aligned} IP(u,a) &= IC(u,a) + QI(u,a) \\ &= 0.8450 + 0.8469 = 1.6919, \\ IP(u,b) &= IC(u,b) + QI(u,b) \\ &= 0.4864 + 0.4755 = 0.9619, \\ IP(u,c) &= IC(u,c) + QI(u,c) \\ &= 0.9053 + 0.9027 = 1.8080, \text{ and} \\ IP(u,d) &= IC(u,d) + QI(u,d) \\ &= 0.7634 + 0.7749 = 1.5383. \end{aligned}$$

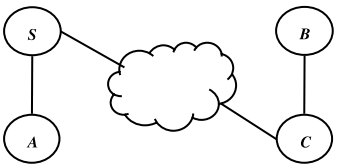


FIGURE 3. Illustration of an autonomous system and the transmission efficiency.

E. TRANSMISSION EFFICIENCY (TE)

This feature considers the transmission distances in a P2P network, as shown in Figure 3. In the figure, peer S must choose a neighbor to dispatch a query to neighbor A or neighbor C. Peers A and S are in the same autonomous system, but peer C belongs to another system. We assume that the transmission path between S and C is much longer than that between S and A. Hence, S will choose the path to A.

The *TE* feature calculates the distances between a peer and its neighbors so that the peer can choose the closest neighbors to deliver a message. Each query peer u computes $SLD_1(u)$, which is the sum of the link-distances of peers that are one hop away from u. Formally,

$$SLD_1(u) = \sum_{v \epsilon N(u)} LD(u,v), \tag{24}$$

where $LD(u, v)$ is the link-distance between u and v.

The *Link-Minus-Score (LMS) of a neighbor v of u* is defined as

$$LMS(u,v) = SLD_1(u) - LD(u,v). \tag{25}$$

According to the above equality, when $LD(u, v)$ increases, the distance between peers u and v also increases; thus, peer v will be assigned a lower score.

Next, we compute $SLMS_1(u)$ (resp. $SLMS_2(u)$), which is the sum of the link-minus-scores of peers that are one (resp. two) hop(s) away from peer u. Formally,

$$SLMS_1(u) = \sum_{v \epsilon N(u)} LMS(u,v) \tag{26}$$

and

$$SLMS_2(u) = \sum_{v \epsilon N(u)} SLMS_1(v). \tag{27}$$

The *transmission efficiency of a neighbor v of peer u* is defined as

$$TE(u,v) = w_1 * \frac{LMS(u,v)}{SLMS_1(u)} + w_2 * \frac{SLMS_1(v)}{SLMS_2(u)}. \tag{28}$$

Based on Eq. (26) and Eq. (27), the scores of peers that are at most two hops away from u are computed. Then, the transmission efficiency of the neighbors of u can be measured and we can determine the quality of each link (u, v).

Example 10: In Figure 2, based on Eq. (24), $SLD_1(u) = LD(u,a) + LD(u,b) + LD(u,c) + LD(u,d) = 103 + 208 + 287 + 50 = 648$; $SLD_1(a) = LD(a,e) + LD(a,f) + LD(a,g) = 141 + 112 + 58 = 311$; $SLD_1(b) = LD(b,h) + LD(b,i) = 211 + 151 = 362$; $SLD_1(c) = LD(c,j) + LD(c,k) + LD(c,l) = 78 + 22 + 106 = 206$; and $SLD_1(d) = LD(d,m) + LD(d,n) = 77 + 28 = 105$. Next, according to Eq. (25), we have $LMS(u,a) = SLD_1(u) - LD(u,a) = 648 - 103 = 545$; $LMS(u,b) = SLD_1(u) - LD(u,b) = 648 - 208 = 440$; $LMS(u,c) = SLD_1(u) - LD(u,c) = 648 - 287 = 361$; and $LMS(u,d) = SLD_1(u) - LD(u,d) = 648 - 50 = 598$. In addition, peer u computes $SLMS_1(u) = LMS(u,a) + LMS(u,b) + LMS(u,c) + LMS(u,d) = 545 + 440 + 361 + 598 = 1,944$ by Eq. (26), and $SLMS_2(u) = SLMS_1(a) + SLMS_1(b) + SLMS_1(c) + SLMS_1(d) = (LMS(a,e) + LMS(a,f) + LMS(a,g)) + (LMS(b,h) + LMS(b,i)) + (LMS(c,j) + LMS(c,k) + LMS(c,l)) + (LMS(d,m) + LMS(d,n)) = ((SLD_1(a) - LD(a,e)) + (SLD_1(a) - LD(a,f)) + (SLD_1(a) - LD(a,g))) + ((SLD_1(b) - LD(b,h)) + (SLD_1(b) - LD(b,i))) + ((SLD_1(c) - LD(c,j)) + (SLD_1(c) - LD(c,k)) + (SLD_1(c) - LD(c,l))) +$

$((SLD_1(d) - LD(d,m)) + (SLD_1(d) - LD(d,n))) = ((311 - 141) + (311 - 112) + (311 - 58)) + ((362 - 211) + (362 - 151)) + ((206 - 78) + (206 - 22) + (206 - 106)) + ((105 - 77) + (105 - 28)) = (170 + 199 + 253) + (151 + 211) + (128 + 184 + 100) + (28 + 77) = 1,501$ by Eq. (27). Then, based on Eq. (28), we have

$$TE(u,a) = 1 * \frac{545}{1,944} + 2 * \frac{170 + 199 + 253}{1,501} = 1.1091,$$

$$TE(u,b) = 1 * \frac{440}{1,944} + 2 * \frac{151 + 211}{1,501} = 0.7087,$$

$$TE(u,c) = 1 * \frac{361}{1,944} + 2 * \frac{128 + 184 + 100}{1,501} = 0.7374,$$

and

$$TE(u,d) = 1 * \frac{598}{1,944} + 2 * \frac{28 + 77}{1,501} = 0.4476.$$

From the results derived in Examples 3, 6, 9, and 10, we obtain the scores of the PA, ES, IP, TE features for peers a, b, c, d in Figure 2(a). They serve as the four columns of the feature matrix of peer u shown in Figure 4.

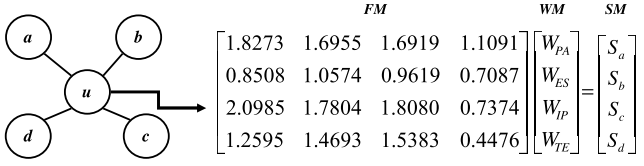


FIGURE 4. **Illustration of Example 10.**

The procedure for constructing the feature matrix FM is presented in Algorithm Constructing_Feature_Matrix.

F. WEIGHT MATRIX

In this section, we introduce the weight matrix, which is incorporated into the feature matrix described in the previous section. It is used to normalize the values of the feature matrix. In the distribution of the data set in the feature matrix, if the values of a feature are widely dispersed, we can set a higher value in the weight matrix to represent the diversity of the feature. Conversely, if the values of a feature are close to each other, we set a lower value in the weight matrix to reduce the influence of the feature. We adopt the mean and standard deviation techniques to achieve this goal.

Let $\{x_1, x_2, x_3, \ldots, x_N\}$ be a data set. First we define the terms.

Definition 1: The *mean*, which is the arithmetic average of a set of values or distribution, is given by

$$\overline{x} = \frac{1}{N} * \sum x_i. \tag{29}$$

Definition 2: The *standard deviation* is a quantity that describes the spread of the data set from the mean. It is given by

$$S = \sqrt{\frac{1}{N-1} * \sum (x_i - \overline{x})^2}. \tag{30}$$

Before computing the standard deviation of each column in the feature matrix, we compute the mean value of each column as follows:

$$\overline{c_m} = \frac{1}{n} * \sum_{j=1}^{n} FM(j,m), \tag{31}$$

Algorithm 1 Constructing_Feature_Matrix(u, w_1, w_2)

for each peer $v \in N(u)$ **do**
  Collect the following information NQ, NR, NF, NFH, NI, NIH, LD from v
  Compute the following terms:
  $SQ_1(u) := \sum_{v \in N(u)} NQ(v)$
  $SR_1(u) := \sum_{v \in N(u)} NR(v)$
  $SF_1(u) := \sum_{v \in N(u)} NF(v)$
  $SFH_1(u) := \sum_{v \in N(u)} NFH(v)$
  $SI_1(u) := \sum_{v \in N(u)} NI(v)$
  $SIH_1(u) := \sum_{v \in N(u)} NIH(v)$
  $SLD_1(u) := \sum_{v \in N(u)} LD(u,v)$
for each peer $v \in N(u)$ **do**
  Collect the following information: SQ_1, SR_1, SF_1, SFH_1, SI_1, SIH_1, SLD_1 from the neighbors of u
  Compute the following terms:
  $SR_2(u) := \sum_{v \in N(u)} SR_1(v)$
  $SF_2(u) := \sum_{v \in N(u)} SF_1(v)$
  $SFH_2(u) := \sum_{v \in N(u)} SFH_1(v)$
  $SI_2(u) := \sum_{v \in N(u)} SI_1(v)$
  $SIH_2(u) := \sum_{v \in N(u)} SIH_1(v)$
for each peer $v \in N(u)$ **do**
  Compute the query-minus-score (QMS) and link-minus-score (LMS):
  $QMS(u,v) := SQ_1(u) - NQ(v)$
  $LMS(u,v) := SLD_1(u) - LD(u,v)$
  Sum the query-minus-scores of the neighbors of u as follows: $SQMS_1(u) := SQMS_1(u) + \sum_{v \in N(u)} QMS(u,v)$
  Sum the link-minus-scores of the neighbors of u as follows: $SLMS_1(u) := SLMS_1(u) + \sum_{v \in N(u)} LMS(u,v)$
  Sum the query-minus-scores of those peers which are two hops away from u as follows: $SQMS_2(u) := SQMS_1(u) + \sum_{v \in N(u)} SQMS_1(v)$
  Sum the link-minus-scores of those peers which are two hops away from u as follows: $SLMS_2(u) := SLMS_1(u) + \sum_{v \in N(u)} SLMS_1(v)$
for each peer $v \in N(u)$ **do**
  Compute the following terms:
  $QF(u,v) := w_1 * \frac{QMS(u,v)}{SQMS_1(u)} + w_2 * \frac{SQMS_1(v)}{SQMS_2(u)}$
  $RF(u,v) := w_1 * \frac{NR(v)}{SR_1(u)} + w_2 * \frac{SR_1(v)}{SR_2(u)}$
  $SC(u,v) := w_1 * \frac{NF(v)}{SF_1(u)} + w_2 * \frac{SF_1(v)}{SF_2(u)}$
  $QS(u,v) := w_1 * \frac{NFH(v)}{SFH_1(u)} + w_2 * \frac{SFH_1(v)}{SFH_2(u)}$
  $IC(u,v) := w_1 * \frac{NI(v)}{SI_1(u)} + w_2 * \frac{SI_1(v)}{SI_2(u)}$
  $QI(u,v) := w_1 * \frac{NIH(v)}{SIH_1(u)} + w_2 * \frac{SIH_1(v)}{SIH_2(u)}$
for each peer $v \in N(u)$ **do**
  Construct the following four columns for the feature matrix FM as follows:
  $PA(u,v) := QF(u,v) + RF(u,v)$
  $ES(u,v) := SC(u,v) + QS(u,v)$
  $IP(u,v) := IC(u,v) + QI(u,v)$
  $TE(u,v) := w_1 * \frac{LMS(u,v)}{SLMS_1(u)} + w_2 * \frac{SLMS_1(v)}{SLMS_2(u)}$
Output the feature matrix *FM*

where n is the number of neighbors of peer u; and $FM(j, m)$ is the value of the entry in the j^{th} row and m^{th} column, i.e., (j, m)-entry, of the feature matrix FM. We compute the standard deviation S_m for $1 \leq m \leq 4$ as follows:

$$S_m = \begin{cases} \sqrt{\frac{1}{n-1} * \sum_{j=1}^{n} (FM(j,m) - \overline{c_m})^2} & \text{if } n \geq 2 \\ 0 & \text{if } n = 1. \end{cases} \quad (32)$$

Then, we define the 4×1 weight matrix WM as

$$WM(m, 1) = \frac{S_m}{\sum_{j=1}^{4} S_j}, \quad (33)$$

where $WM(m, 1)$ is the weight of the $(m, 1)$-entry.

Example 11: Based on Eq. (33) and the feature matrix constructed in Example 10, the weight matrix is $[0.3662, 0.2111, 0.2450, 0.1777]^T$.

The procedure for constructing the weight matrix is presented in Algorithm Constructing_Weight_Matrix.

Algorithm 2 Constructing_Weight_Matrix (FM)

1: **for** each column m: $= 1$ to 4 **do** /* Compute the mean value $\overline{c_m}$ and the standard deviation S_m for column m */
2: Compute $\overline{c_m} := \frac{1}{n} * \sum_{j=1}^{n} FM(j, m)$
3: **if** n: $= 1$ **then**
4: S_m: $= 0$
5: **else**
6: $S_m := \sqrt{\frac{1}{n-1} * \sum_{j=1}^{n} (FM(j,m) - \overline{c_m})^2}$
7: **for** each column m: $= 1$ to 4 **do** /* Compute the four elements of the weight matrix */
8: $WM(m, 1) := \frac{S_m}{\sum_{j=1}^{4} S_j}$
9: **Output the weight matrix** WM

G. SCORING MATRIX

After executing Algorithm Constructing_Feature_Matrix and Algorithm Constructing_Weight_Matrix, each query peer u constructs an $n \times 1$ scoring matrix (SM) by the multiplication of the feature matrix and the weight matrix as follows:

$$SM(m, 1) = \sum_{k=1}^{4} (FM(m, k) * WM(k, 1)), \quad (34)$$

where $SM(m, 1)$ for $1 \leq m \leq n$ is the $(m, 1)$-entry of SM and n is the number of neighbors of u.

Based on the scoring matrix, each query peer u can assess the capability of each of its neighbors. To deliver query messages, peer u selects k influential neighbors according to the top-k values in the scoring matrix. The proper value of k is determined by the parameters of a P2P network e.g., the scale of the network, the number of neighbors of a query peer, and the requirement of a query peer. (We explain how the value of k is determined in Section IV-D3.) Moreover, it has been shown that, in a real P2P network, the number of neighbors of a query peer is usually 6 at most [17], [25], [51], which implies that the scoring matrix should not be too large. As a result, the performance of the proposed search mechanism can be optimized because the memory requirement is small.

Example 12: By multiplying the feature matrix and the weight matrix derived in Example 10 and Example 11 respectively, we construct the scoring matrix shown in Figure 5.

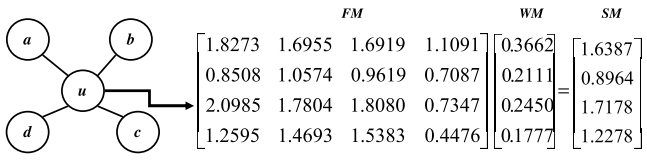


FIGURE 5. Illustration of Example 12.

IV. PERFORMANCE EVALUATION

In this section, we present the experimental results, which demonstrate the effectiveness of the proposed search mechanism SMF. We construct two types of network topology, a physical topology and a logical topology. The physical topology is comprised of the real connections in a large-scale network; and the logical topology is the P2P transmission layer, which is built on top of the physical topology connections. All the peers in the logical network are chosen from the physical network. We compare the search efficiency of SMF with that of other search mechanisms, namely Flooding (FL), Random Walk (RW), and Multiple Random Walk (MRW) in terms of the following four criteria:

- **Traffic Cost**: the average number of messages sent per query; it is used to measure the search efficiency.
- **Response Time**: the query response time is the interval between the time a query is initiated and the time the result is received when each query explores the matching response in the P2P network.
- **Query Success Rate**: the percentage of queries that receive at least one response during the search process.
- **Searching Hop in Query Hit**: the average number of steps required to send a query if the query is responded to in our experiment.

A. PARAMETER SETTINGS

In the experimental environment, we create five physical topologies, each containing 50,000 peers. Logical topologies built on top of the physical layer contains 5,000, 10,000, and 20,000 peers respectively. The experimental parameters are the same as those in [25] and [51]: the average degree is between 3 and 9 and the TTL is set at 7. By considering the dynamic property of unstructured P2P systems, we also develop a dynamic environment, which means that peers join (leave) a P2P network in a random fashion. Queries are produced by each peer with equal probability. In addition, to compare different search mechanisms and different parameters of SMF, the experiments are performed on networks with 1,000, 5,000, 10,000, and 20,000 peers.

At the beginning of the experiment, every peer in the virtual network has a randomly initialized file sharing,

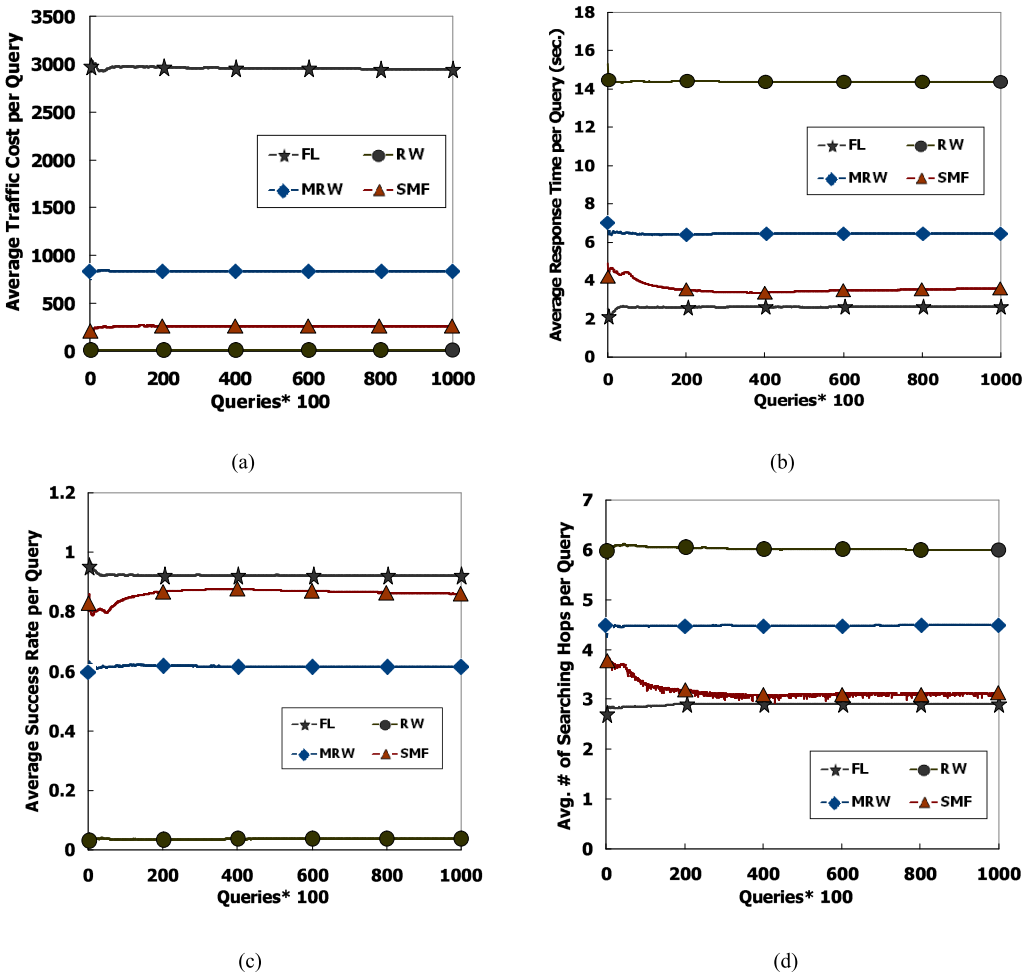


FIGURE 6. Comparison of the following performance metrics in static environments: (a) traffic cost, (b) response time, (c) success rate, and (d) average number of search hops.

peer connectivity topology, and index routing table. Initially, all peers in the logical network generate query messages randomly. The number of queried neighbors in each query process of MRW and SMF is between 2 and 5. The experiment stops when a preset number of queries (10,000, 50,000, 100,000, and 200,000 queries) have been sent. For ease of presentation, we only consider the experimental results based on the following settings: the number of peers in the logical topology is 20,000; the average node degree in the P2P network is 6; the number of query neighbors in both MRW and SMF is 2; and the total number of queries sent is 100,000.

B. EXPERIMENTS IN A STATIC ENVIRONMENT

In this subsection, we discuss the effectiveness of SMF in static environments, which means the peers do not join (leave) the network in a random manner. The parameter k used in RW and MRW is set at 2 because, if k is larger, the mechanism would be similar to the flooding approach in which a query peer sends each message to all of its neighbors. Compared with FL and MRW, SMF reduces the traffic cost significantly, as shown in Figure 6(a). Note that RW has the lowest traffic cost because it only selects one neighbor at random to send a message without considering the success rate. Compared with FL, SMF reduces the traffic cost significantly. Thus, we achieve the goal of minimizing the traffic cost when searching for files in a P2P network. Moreover, in contrast to MRW, which chooses a fixed number of neighbors, k, at random to send messages without any strategy, SMF only selects influential neighbors and thereby optimizes the search performance with respect to the traffic cost in static environments.

Figure 6(b) shows that FL achieves the shortest query response time. The proposed SMF outperforms RW and MRW in terms of the response time. In contrast to FL, which sends a query message to all of a query peer's neighbors, SMF only sends a message to a small number of such neighbors. Since FL uses a large amount of resources when searching for matching responses, it achieves the best performance in terms of the response time. However, it also incurs a high traffic overhead. The SMF is effective because it only selects influential neighbors to send messages, yet its performance is still as good as that of FL.

Next, we analyze the success rates of the compared methods. Figure 6(c) shows that SMF obtains response for about 85 percent of query messages. Although FL gets responses

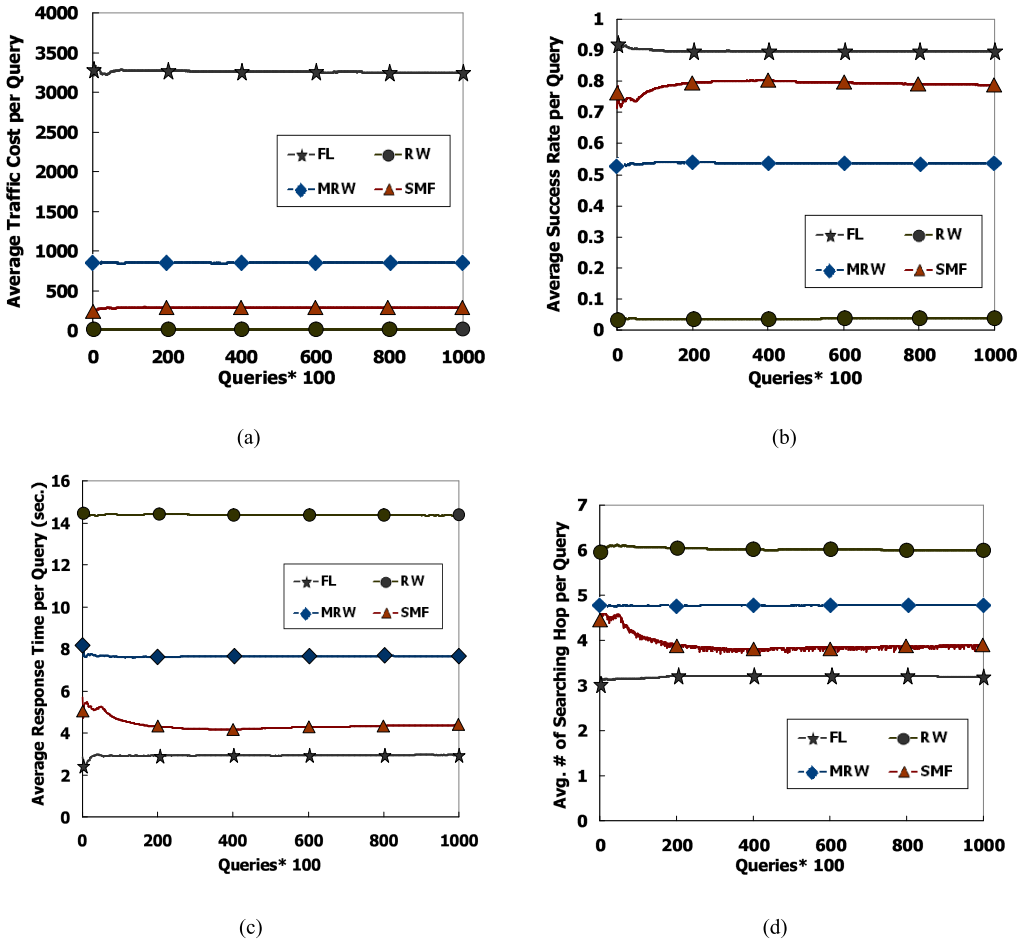


FIGURE 7. Comparison of the following four performance metrics in dynamic environments: (a) traffic cost, (b) success rate, (c) response time, and (d) number of search hops.

to about 90 percent of the queries, its traffic overhead is high because the messages are sent to all of a peer's neighbors. Therefore, except for FL, the compared search mechanisms reduce the traffic cost at the expense of a lower success rate. However, the reduction in SMF's success rate is smaller than that of RW and MRW.

Figure 6(d) shows the average number of search hops. In a static environment, SMF requires about three hops to obtain responses. The performance is similar to that of FL, but SMF does not need to send queries to all the neighbors of a query peer. Moreover, as shown in the figure, SMF obtains query responses in short hops. Overall, the experimental results demonstrate that SMF performs well in static environments.

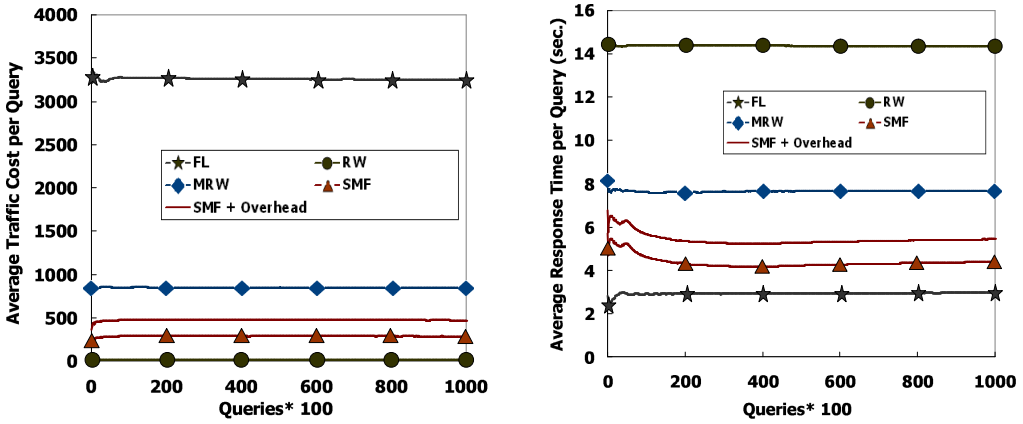
C. EXPERIMENTS IN DYNAMIC ENVIRONMENTS

As shown in Figures 7(a) and 7(b), SMF is more effective than FL, which incurs an enormous traffic overhead in order to increase the number of opportunities to match files during the search process. Meanwhile, Figure 7(b) shows that SMF also has a higher probability of obtaining responses. Thus, SMF has the advantages of FL, but its traffic overhead is much lower. Comparison of MRW and SMF shows that SMF's search cost is lower, even though it selects the same number of neighbors to send a query. Finally, although RW has the lowest traffic overhead, its success rate is the lowest. Thus, overall, SMF achieves an effective search performance.

In Figure 7(c), the traditional search mechanism FL has the shortest response time. This is because it delivers queries to all of a peer's neighbors. SMF's response time is a little higher than that of FL, but it reduces the traffic overhead significantly. Although the number of neighbors selected by SMF to send a query is the same as that of MRW in the experiment, SMF performs better than MRW. Hence, SMF can achieve a shorter query response time in dynamic environments.

Figure 7(d) shows that SMF's performance is a little worse than that of FL, but it outperforms the other two search mechanisms. Although SMF does not send messages to all the neighbors of a query peer, it can still get query responses within an acceptable search scope. The experiment results show that SMF can obtain matching files by setting the d-collected scope at $d = 4$. We discuss this aspect in Section IV-D2.

In addition to the features compared above, SMF achieves a better performance through the warm-up phase avoiding. As shown in Figures 7(a)–7(d), SMF's performance is not



(a) (b)

FIGURE 8. **Comparison of the following performance metrics under SMF with the construction overhead: (a) traffic cost and (b) response time.**

satisfactory in the initial stage; however, as the number of queries increases, the search performance improves. The reason is that the matrix form yields more precise computations as the number of queries increases. Since communications in a real P2P network are continuous, we can obtain accurate information about each peer via the matrix.

D. ANALYSIS OF THE SMF

In this section, we discuss some critical issues related to SMF, namely the matrix construction overhead, the proper settings for the parameter values, and the search performance in different-sized networks.

1) MATRIX CONSTRUCTION OVERHEAD

We add the *construction overhead* to SMF to compare its performance with that of the other search mechanisms. As shown in Figure 8(a), SMF with the construction overhead reduces the traffic cost of FL by about 85 percent, yet it still achieves a good success rate in the query process. The experimental results show that SMF trades a little of the success rate to achieve a substantial improvement in the traffic cost. Meanwhile, SMF reduces the traffic cost of MRW by about 45 percent, despite the matrix construction overhead. Hence, the construction overhead incurred by collecting the feature information in a whole P2P network is acceptable. It appears that considering influential neighbors to send a query message is more effective than choosing them at random.

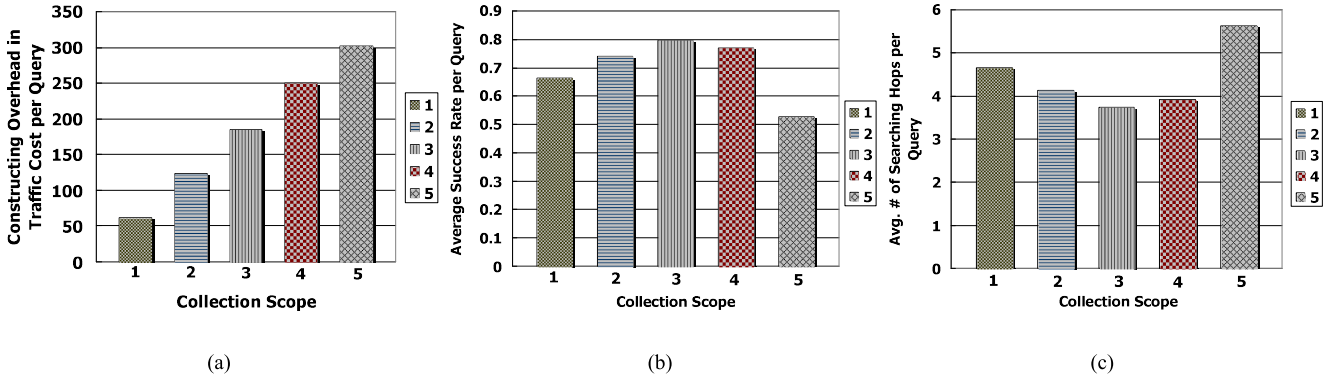
As shown in Figure 8(b), SMF reduces the response time of MRW by about 43 percent. In SMF, when a peer needs to choose a neighbor to deliver a query, it computes the distances of the peer's neighbors in the *TE* of the feature matrix. If the distance between a peer and a neighbor is too great, the neighbor may not be able to deliver the query in the shortest possible time, so it will not be selected. SMF with the construction overhead improves the response time by about 30 percent with the parameter settings and simulation environment described in Subsection IV-C. Although FL achieves a better performance than SMF in terms of the response time, SMF's traffic overhead is much lower than that of FL. Unlike FL, which sends queries to all the neighbors of a query peer, SMF only sends queries to specific neighbors to exploit matching files. As a result, FL has many more paths to respond to the source peer than SMF. Therefore, although the SMF is less efficient than FL, it reduces the traffic overhead substantially.

2) SETTING PROPER VALUES FOR THE FEATURE COLLECTION SCOPE

In this subsection, we discuss the proper setting for the value d of the feature collection scope mentioned in Section III-A. Figures 9(a)–9(c) show that when $d = 5$, the construction overhead increases. Moreover, when $d = 4$, both the success rate and the average number of search hops are better than when $d = 1$ or $d = 2$. However, if $d = 5$, the success rate and the average number of search hops are not as good as those under the other settings. To verify the above observation, we refer to the results in Figure 7(d), which show that the average number of search hops is set at 4. If the peers collect information with $d = 5$, the scope may include many useless messages, which will increase the computational overhead. Hence, we only consider cases where d is between 1 and 4. When $d = 1$, the success rate is only about 60 percent, so 1 is not suitable for SMF. In contrast, under $d = 2$ and $d = 4$, the success rates are about 75 percent, but the construction overhead of $d = 4$ is higher than that of $d = 2$. Hence, $d = 2$ is an appropriate setting for SMF. On the other hand, when $d = 3$, SMF has the highest success rate, the lowest number of search hops, and an acceptable matrix construction overhead. Hence, we conclude that either $d = 2$ or $d = 3$ is an appropriate value setting for the feature collection scope.

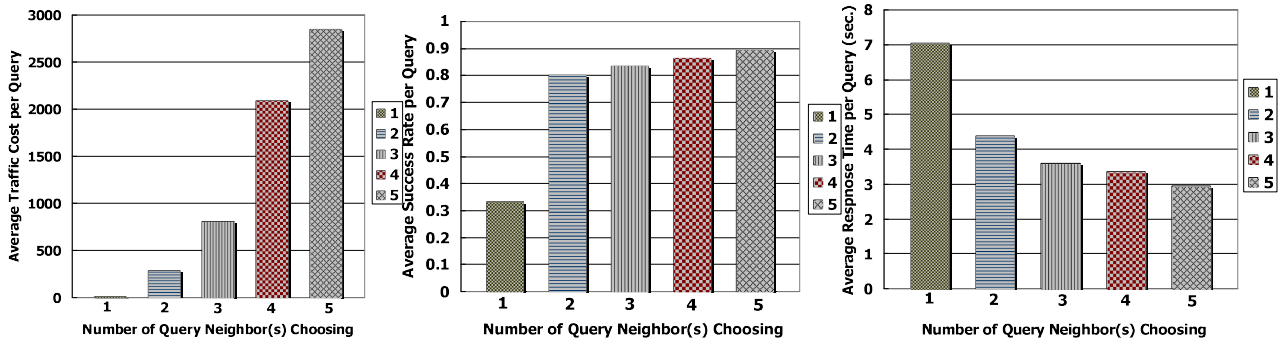
3) THE NUMBER OF QUERIED NEIGHBORS

Given the scoring matrix, how many neighbors should each query peer choose to send messages in order to optimize



(a) (b) (c)

FIGURE 9. The feature collection scope and construction overhead under the following performance metrics: (a) traffic cost, (b) success rate, and (c) average number of search hops.



(a) (b) (c)

FIGURE 10. Comparison of the following performance metrics when choosing the number of query neighbors: (a) traffic cost, (b) success rate, and (c) response time.

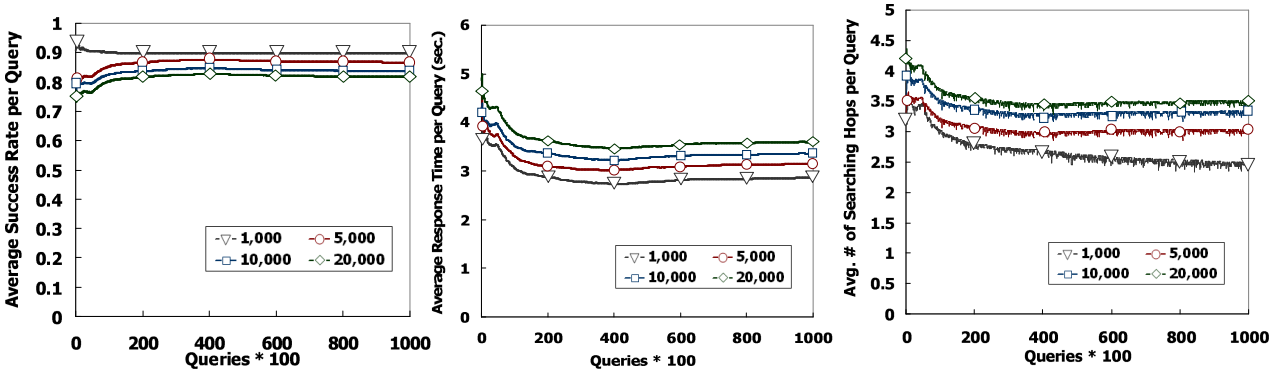
TABLE 1. Comparison of the search performance with different hop weight settings.

(w_1, w_2)	Avg. Traffic Cost	Avg. Response Time	Success Rate	Avg. # of Search Hop(s)
$(1, 2)$	282.24	4.09	0.8172	3.43
$(1, 3)$	270.69	3.96	0.8209	3.36
$(1, 4)$	267.04	3.86	0.8368	3.33
$(2, 2)$	286.25	4.51	0.7879	3.66
$(2, 1)$	288.06	4.53	0.7885	3.70
$(3, 1)$	296.04	5.41	0.7183	4.68
$(4, 1)$	307.75	6.35	0.6396	5.41

the search performance? As shown in Figure 10(a), when the number of selected neighbors increases, the traffic cost increases sharply. Hence, if a query peer chooses too many neighbors to send messages, SMF will select too many powerless neighbors to send the messages and thereby increase the traffic overhead. In Figure 10(b), when the number of neighbors selected is set at 1, SMF still achieves a higher success rate than RW, as shown in Figure 7(b). When the number of neighbors chosen is set at 1, the value is not good for the number of neighbors chosen because the success rate is only about 30 percent. When the number is set higher than 2, the success rate is good, but at 4 or 5, SMF's traffic overhead is too high. On the other hand, as shown in Figure 10(c), SMF achieves the best query response time if only 1 neighbor is chosen. From the above observations, either 2 or 3 is an appropriate setting for the number of neighbors chosen by a query peer.

4) PROPER WEIGHT SETTING

In this subsection, we discuss the hop weight setting. According to the construction of the feature matrix described in Section III, if $w_1 > w_2$, one-hop collected information is more important than two-hop collected information. Conversely, if $w_1 < w_2$, two-hop information is more important. Table 1 shows the search performance of SMF with different hop weight settings. The results show that SMF's search performance is better when $w_2 > w_1$. We can conclude that the wider search scope of SMF, the larger will be the



(a) (b) (c)

FIGURE 11. Comparison of the following performance metrics of SMF on different network scales: (a) success rate, (b) response time, and (c) average number of search hops.

number of peers whose information and features need to be collected and computed. As shown in the table, the search performance is optimal when $(w_1, w_2) = (1, 4)$; and it is the least efficient when w_1 is much higher than w_2.

5) SEARCH PERFORMANCE IN DIFFERENT SIZED NETWORKS

In this subsection, we discuss the effectiveness of SMF in different sized networks. Figure 11(a) shows that SMF achieves a good search performance in different network scales. As the size of the network increases, the success rate (i.e., the file matching rate) declines. Nevertheless, SMF's success rate is still about 80 percent. Figures 11(b) and 11(c) show the response time and the average number of search hops respectively. We observe that SMF achieves shorter response times as the network scale increases. Hence, SMF can match files successfully in a small search scope under different network scales. The experimental results indicate that SMF is also suitable for large-scale P2P networks.

V. CONCLUDING REMARKS

We have performed simulations of the proposed SMF search mechanism in static and dynamic environments. The results demonstrate that SMF can reduce the traffic overhead significantly, achieve shorter query responded times, and maintain a high success rate. Specifically, SMF performs more than 80 percent better than the flooding approach in terms of the traffic overhead. Compared to the multiple random walk approach, SMF's response times and success rate are 40 percent and 20 percent better respectively. The experimental results also show that each peer can determine the capability of each neighbor peer and send messages to the appropriate number of its neighbors to avoid redundant messages. Therefore, SMF is an effective search mechanism for P2P networks.

SMF can also be utilized in different applications, e.g., to adjust the connections for clustering influential peers, such as the clustering approach proposed in [18], [24], [39], [58], and [60], or to reconstruct a topology, like the minimal spanning tree technique used in [30], [31], and [61]. By utilizing the statistical matrix form, we can obtain more precise information and further improve the search performance in P2P networks.

REFERENCES

[1] (2011). *BitTorrent*. [Online]. Available: http://www.bittorrent.com/
[2] (2013). *Chord*. [Online]. Available: http://www.pdos.lcs.mit.edu/chord/
[3] (2011). *eMule*. [Online]. Available: http://www.emule-project.net/
[4] (2015). *FrostWire*. [Online]. Available: http://www.frostwire.com/
[5] (2008). *Gnutella*. [Online]. Available: http://rfc-gnutella.sourceforge.net/
[6] (2011). *KaZaa*. [Online]. Available: http://www.kazaa.com/
[7] (2010). *LimeWire*. [Online]. Available: http://www.limewire.com/
[8] (2009). *Pastry*. [Online]. Available: http://www.freepastry.org/
[9] E. Adar and B. A. Huberman, "Free riding on Gnutella," *First Monday*, vol. 5, nos. 10–12, pp. 1–22, Oct. 2000.
[10] R. Ahmed and R. Boutaba, "Plexus: A scalable peer-to-peer protocol enabling efficient subset search," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 130–143, Feb. 2009.
[11] S. Bianchi, P. Felber, and M. G. Potop-Butucaru, "Stabilizing distributed R-trees for peer-to-peer content routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 8, pp. 1175–1187, Aug. 2010.
[12] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P systems scalable," in *Proc. ACM SIGCOMM*, 2003, pp. 407–418.
[13] E. Chavez, M. Graff, G. Navarro, and E. S. Tellez, "Near neighbor searching with K nearest references," *J. Inf. Syst.*, vol. 51, pp. 43–61, Jul. 2015.
[14] K. Chen *et al.*, "Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 1021–1036, Apr. 2014.
[15] G. Chen, C. P. Low, and Z. Yang, "Enhancing search performance in unstructured P2P networks based on users' common interest," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 6, pp. 821–836, Jun. 2008.
[16] W.-M. Chen and K.-C. Liu, "Randomized search strategy for unstructured P2P networks," *IEICE Trans. Commun.*, vol. E95-B, no. 1, pp. 289–292, 2012.
[17] Y.-M. Chiu and D. Y. Eun, "On the performance of content delivery under competition in a stochastic unstructured peer-to-peer network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 10, pp. 1487–1500, Oct. 2010.
[18] S. Datta, C. R. Giannella, and H. Kargupta, "Approximate distributed K-means clustering over a peer-to-peer network," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 10, pp. 1372–1388, Oct. 2009.
[19] S. S. Dhillon and P. Van Mieghem, "Searching with multiple random walk queries," in *Proc. IEEE 18th Int. Symp. Personal, Indoor Mobile Radio Commun.*, Sep. 2007, pp. 1–5.
[20] P. Fonseca, R. Rodrigues, A. Gupta, and B. Liskov, "Full-information lookups for peer-to-peer overlays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 9, pp. 1339–1351, Sep. 2009.

[21] R. Gaeta and M. Sereno, "Generalized probabilistic flooding in unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2055–2062, Dec. 2011.
[22] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Mar. 2004, pp. 1–12.
[23] H. Guclu and M. Yuksel, "Limited scale-free overlay topologies for unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 667–679, May 2009.
[24] K. M. Hammouda and M. S. Kamel, "Hierarchically distributed peer-to-peer document clustering and cluster summarization," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 681–698, May 2009.
[25] M. Hefeeda and B. Noorizadeh, "On the benefits of cooperative proxy caching for peer-to-peer traffic," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 998–1010, Jul. 2010.
[26] M. Hefeeda, C.-H. Hsu, and K. Mokhtarian, "Design and evaluation of a proxy cache for peer-to-peer traffic," *IEEE Trans. Comput.*, vol. 60, no. 7, pp. 964–977, Jul. 2011.
[27] P. Hieungmany, T. Souma, and S. Shioda, "Directional-random-walk-based contents search for unstructured P2P systems," *IEICE Trans. Commun.*, vol. J98-B, no. 2, pp. 132–140, 2015.
[28] H. Hoang-Van, Y. Shinozaki, T. Miyoshi, and O. Fourmaux, "A router-aided hierarchical P2P traffic localization based on variable additional delay insertion," *IEICE Trans. Commun.*, vol. E97-B, no. 1, pp. 29–39, 2014.
[29] D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on Gnutella revisited: The bell tolls?" *IEEE Distrib. Syst. Online*, vol. 6, no. 6, pp. 1–18, Jun. 2005.
[30] H.-C. Hsiao and C.-H. He, "A tree-based peer-to-peer network with quality guarantees," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 8, pp. 1099–1110, Aug. 2008.
[31] H.-C. Hsiao, H. Liao, and C.-C. Huang, "Resolving the topology mismatch problem in unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 11, pp. 1668–1681, Nov. 2009.
[32] H.-C. Hsiao, H. Liao, and P.-S. Yeh, "A near-optimal algorithm attacking the topology mismatch problem in unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 983–997, Jul. 2010.
[33] H.-C. Hsiao, Y.-C. Lin, and H. Liao, "Building small-world peer-to-peer networks based on hierarchical structures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 7, pp. 1023–1037, Jul. 2009.
[34] S. Jiang, L. Guo, X. Zhang, and H. Wang, "LightFlood: Minimizing redundant messages and maximizing scope of peer-to-peer search," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 601–614, May 2008.
[35] M. Karakaya, I. Korpeoglu, and O. Ulusoy, "Free riding in peer-to-peer networks," *IEEE Internet Comput.*, vol. 13, no. 2, pp. 92–98, Mar./Apr. 2009.
[36] A. Kumar, J. Xu, and E. W. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks," in *Proc. 24th IEEE Int. Conf. Comput. Commun.*, vol. 2. Mar. 2005, pp. 1162–1173.
[37] H. Lee and A. Nakao, "A feasibility study of P2P traffic localization through network delay insertion," *IEICE Trans. Commun.*, vol. E95-B, no. 11, pp. 3464–3471, 2012.
[38] Y. Li, D. Gruenbacher, and C. Scoglio, "Evaluating stranger policies in P2P file-sharing systems with reciprocity mechanisms," *Comput. Netw.*, vol. 56, no. 4, pp. 1470–1485, 2012.
[39] Z. Li, J. Wu, J. Xie, T. Zhang, G. Chen, and Y. Dai, "Stability-optimal grouping strategy of peer-to-peer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2079–2087, Dec. 2011.
[40] T. Lin, P. Lin, H. Wang, and C. Chen, "Dynamic search algorithm in unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 654–666, May 2009.
[41] Y. Liu, J. Han, and J. Wang, "Rumor riding: Anonymizing unstructured peer-to-peer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 464–475, Mar. 2011.
[42] X. Liu, Y. Liu, and L. Xiao, "Improving query response delivery quality in peer-to-peer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 11, pp. 1335–1347, Nov. 2006.
[43] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang, "Location awareness in unstructured peer-to-peer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 163–174, Feb. 2005.
[44] Y. Liu, L. Xiao, and L. M. Ni, "Building a scalable bipartite P2P overlay network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 9, pp. 1296–1306, Sep. 2007.
[45] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "AOTO: Adaptive overlay topology optimization in unstructured P2P systems," in *Proc. IEEE GLOBECOM*, vol. 7. Dec. 2003, pp. 4186–4190.
[46] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. 16th Int. Conf. Supercomput.*, 2002, pp. 84–95.
[47] T. Nakashima and S. Fujita, "Tree-based consistency maintenance scheme for peer-to-peer file sharing of editable contents," *IEICE Trans. Inf. Syst.*, vol. E97-D, no. 12, pp. 3033–3040, 2014.
[48] S. Patro and Y. C. Hu, "Transparent query caching in peer-to-peer overlay networks," in *Proc. 17th Int. Parallel Distrib. Process. Symp.*, Apr. 2003, pp. 1–10.
[49] K. K. Ramachandran and B. Sikdar, "A queuing model for evaluating the transfer latency of peer-to-peer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 3, pp. 367–378, Mar. 2010.
[50] S. C. Rhea and J. Kubiatowicz, "Probabilistic location and routing," in *Proc. IEEE Int. Conf. Comput. Commun.*, vol. 3. Jun. 2002, pp. 1248–1257.
[51] R. Matei, I. Foster, and A. Iamnitchi, "Mapping the Gnutella network," *IEEE Internet Comput.*, vol. 6, no. 1, pp. 50–57, Jan./Feb. 2002.
[52] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. 18th IFIP/ACM Int. Conf. Distrib. Syst. Platforms*, 2001, pp. 1–22.
[53] K. Saito and Y. Takano, "Distributed hash tables adapting to the conditions of the real world," *IEICE Trans. Inf. Syst.*, vol. J96-D, no. 6, pp. 1433–1446, 2013.
[54] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. Multimedia Comput. Netw.*, 2002, pp. 1–15.
[55] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun.*, vol. 3. Mar./Apr. 2003, pp. 2166–2176.
[56] H. Stark and J. G. Brankov, "Estimating the standard deviation from extreme Gaussian values," *IEEE Signal Process. Lett.*, vol. 11, no. 3, pp. 320–322, Mar. 2004.
[57] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *Proc. ACM SIGCOMM*, 2001, pp. 149–160.
[58] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 377–390, Apr. 2009.
[59] Y. Takahashi, T. Izumi, H. Kakugawa, and T. Masuzawa, "An efficient index dissemination in unstructured peer-to-peer networks," *IEICE Trans. Inf. Syst.*, vol. E91-D, no. 7, pp. 1971–1981, 2008.
[60] S. Tewari and L. Kleinrock, "Optimal search performance in unstructured peer-to-peer networks with clustered demands," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 84–95, Jan. 2007.
[61] C. Theoharatos, G. Economou, and S. Fotopoulos, "Color edge detection using the minimal spanning tree," *Pattern Recognit.*, vol. 38, no. 4, pp. 603–606, 2005.
[62] C. Tian, H. Jiang, X. Liu, and W. Liu, "Revisiting dynamic query protocols in unstructured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 160–167, Jan. 2012.
[63] D. Tsoumakos and N. Roussopoulos, "Analysis and comparison of P2P search methods," in *Proc. ACM 1st Int. Conf. Scalable Inf. Syst.*, 2006, Art. ID 25.
[64] C. Wang, L. Xiao, Y. Liu, and P. Zheng, "DiCAS: An efficient distributed caching mechanism for P2P systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 10, pp. 1097–1109, Oct. 2006.
[65] C. Wang and L. Xiao, "An effective P2P search scheme to exploit file sharing heterogeneity," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 145–157, Feb. 2007.
[66] H. Wang and T. Lin, "On efficiency in searching networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, vol. 2. Mar. 2005, pp. 1490–1501.
[67] L. Xiao, Y. Liu, and L. M. Ni, "Improving unstructured peer-to-peer systems by adaptive connection establishment," *IEEE Trans. Comput.*, vol. 54, no. 9, pp. 1091–1103, Sep. 2005.
[68] X. Xiao, Q. Zhang, Y. Shi, and Y. Gao, "How much to share: A repeated game model for peer-to-peer streaming under service differentiation incentives," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 288–295, Feb. 2012.
[69] B. Yang and H. Garcia-Molina, "Efficient search in peer-to-peer networks," in *Proc. 16th Annu. ACM Symp. Parallel Algorithms Archit.*, 2004, pp. 271–272.

[70] Z. Yang, J. Tian, B. Y. Zhao, W. Chen, and Y. Dai, "Protector: A probabilistic failure detector for cost-effective peer-to-peer storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1514–1527, Sep. 2011.
[71] M. Yang and Y. Yang, "An efficient hybrid peer-to-peer system for distributed data sharing," *IEEE Trans. Comput.*, vol. 59, no. 9, pp. 1158–1171, Sep. 2010.
[72] K. Yokota, T. Nakagawa, T. Isogai, T. Asaka, and T. Takahashi, "Clustering for distributing cached contents in P2P file sharing application," *IEICE Trans. Commun.*, vol. J95-B, no. 2, pp. 178–187, 2012.
[73] Y. B. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. CSD-01-1141, 2001, pp. 1–27.
[74] B. Q. Zhao, J. C. Lui, and D.-M. Chiu, "A mathematical framework for analyzing adaptive incentive protocols in P2P networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 367–380, Apr. 2012.
[75] Z. Zhu, P. Kalnis, and S. Bakiras, "DCMP: A distributed cycle minimization protocol for peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 3, pp. 363–377, Mar. 2008.

CHIA-HUNG LIN received the B.S. degree from the Department of Information Management, National Taiwan University of Science and Technology, Taiwan, in 2001, and the M.S. degree in computer science and information engineering from National Cheng Kung University, Taiwan, in 2004, where he is currently pursuing the Ph.D. degree in computer science and information engineering. His research interests include parallel and distributed computing, design and analysis of algorithms, human–machine interface design, and machine learning.

JING-JIA ZSENG received the B.S. degree from the Department of Computer Science and Information Engineering, National Changhua University of Education, Taiwan, in 2008, and the M.S. degree in computer science and information engineering from National Cheng Kung University, Taiwan, in 2010. His research interests include parallel and distributed computing, peer-to-peer computing, and cloud computing.

SUN-YUAN HSIEH received the Ph.D. degree in computer science from National Taiwan University, Taipei, Taiwan, in 1998. He then served the compulsory two-year military service. From 2000 to 2002, he was an Assistant Professor with the Department of Computer Science and Information Engineering, National Chi Nan University. In 2002, he joined the Department of Computer Science and Information Engineering, National Cheng Kung University, where he is currently a Distinguished Professor. He is a fellow of the British Computer Society. He was a recipient of many awards, including the K. T. Lee Research Award in 2007, the President's Citation Award (the American Biographical Institute) in 2007, the Engineering Professor Award of the Chinese Institute of Engineers (Kaohsiung Branch) in 2008, the National Science Council's Outstanding Research Award in 2009, the IEEE Outstanding Technical Achievement Award (the IEEE Tainan Section) in 2011, the Outstanding Electronic Engineering Professor Award of the Chinese Institute of Electrical Engineers in 2013, and the Outstanding Engineering Professor Award of the Chinese Institute of Engineers in 2014.

Dr. Hsieh is also an Experienced Editor with editorial services to a number of journals, including serving as an Associate Editor of the IEEE Access, *Theoretical Computer Science* (Elsevier), *Discrete Applied Mathematics* (Elsevier), the *Journal of Supercomputing* (Springer), the *International Journal of Computer Mathematics* (Taylor & Francis Group), *Discrete Mathematics, Algorithms and Applications* (World Scientific), *Fundamenta Informaticae* (Polish Mathematical Society), and the *Journal of Interconnection Networks* (World Scientific). In addition, he has served on the organization committee and/or program committee of several dozens of international conferences in computer science and computer engineering. His current research interests include design and analysis of algorithms, fault-tolerant computing, bioinformatics, parallel and distributed computing, and algorithmic graph theory.