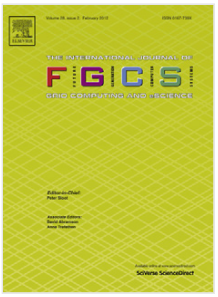
Accepted Manuscript



Distance-aware bloom filters: Enabling collaborative search for efficient
resource discovery

Yiming Zhang, Ling Liu

PII: S0167-739X(12)00163-X
DOI: 10.1016/j.future.2012.08.007
Reference: FUTURE 2260

To appear in: *Future Generation Computer Systems*

Received date: 9 December 2011
Revised date: 8 August 2012
Accepted date: 22 August 2012

Please cite this article as: Y. Zhang, L. Liu, Distance-aware bloom filters: Enabling collaborative search for efficient
resource discovery, *Future Generation Computer Systems* (2012), doi:10.1016/j.future.2012.08.007

Distance-aware bloom filters: Enabling collaborative search for efficient resource discovery

Yiming Zhang[*], Ling Liu[†]

[*]School of Computer, National University of Defense Technology, Changsha 410073, China
[†]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, USA

Abstract

Resource discovery in large-scale Peer-to-Peer (P2P) networks is challenging due to lack of effective methods for guiding queries. Based on the observation that the effectiveness of P2P resource discovery is determined by the utilization of hints, i.e., a summary of where the resources are, scattered in the network, in this paper we propose the distance-aware bloom filters (DABF) that disseminate hint information to faraway nodes by decaying BFs with different deterministic masks. Based on DABF, we design a novel Collaborative P2P Search (CPS) mechanism, which supports intelligent message behaviours including suspend, resume, terminate, move, reside, dispatch, notify and order. The effectiveness of our proposals is demonstrated through theoretical analysis and extensive simulations, in which we observed a remarkable reduction in search latency over previous approaches.

KEY WORDS: Peer-to-Peer; distance-aware bloom filters (DABF); collaborative search; information aggregation; resource discovery

[*]Correspondence to: Yiming Zhang, National Laboratory for Parallel and Distributed Processing (PDL); School of Computer, National University of Defense Technology (NUDT). Changsha 410073, China.
[†]E-mail: ymzhang@nudt.edu.cn.

1. INTRODUCTION

In recent years Peer-to-Peer (P2P) computing [1] has emerged as a milestone in distributed computing. It enables large-scale sharing of resources geographically distributed and belonging to unrelated administrative domains [2].

Efficient *resource discovery* in large-scale P2P networks is a challenging problem. Early schemes such as flooding and Random Walk produce large amounts of redundant messages and consume much bandwidth [3]. This has motivated a lot of studies about bandwidth-efficient P2P resource discovery such as Biased Random Walks [1], ARW [4], *k*-RW [5], and so on. However, all of these proposals retain the "blind" nature of Flooding and Random Walk, and thus suffer from a long delay problem in large-scale networks.

Intuitively, the efficiency of P2P resource discovery would be largely improved if queries can be guided by some sort of hints. Recently a lot of "informed" P2P schemes have been proposed. Examples include SQR [6], PlanetP [7], and Attenuated search [8]. These schemes propagate hint information (i.e., a summary of where the resources are) to guide queries, which is usually compressed based on the bloom filters (BF) [9]. Clearly, the performance of "informed" schemes is tightly coupled with the maximum distance (number of hops) that the hints can be propagated, and within this distance queries can be effectively guided. However, current P2P schemes can only propagate the hints in a very small neighborhood, which apparently takes little effect in large-scale networks [6].

Based on the observation that the effectiveness of P2P resource discovery is determined by the usefulness of hints scattered in the network, in this paper we propose a novel structure called distance-aware bloom filters (DABF) and design a DABF-based Collaborative P2P Search (CPS) mechanism for efficient P2P resource discovery.

(i) To propagate the hints as far as possible with appropriate bandwidth consumption, DABF divides traditional BF vectors into subvectors and disseminates the fragments in a bandwidth-efficient manner. Here the term "bandwidth-efficient" has two meanings. First, the subvector sent to a neighbor is proportional to that neighbor's available bandwidth, letting a more capable neighbor receive a bigger chunk of the vector. Second, DABF sends only partial information to each of its neighbors, which alleviates the problem of huge bandwidth consumption in traditional BF techniques.

(ii) To effectively use the partial resource information propagated by DABF to guide queries, we design a collaborative search algorithm, in which query

messages aggregate the fragment hints scattered in the network and collaboratively make intelligent decisions, e.g., whether the search in certain regions should be enhanced, weakened or even terminated. By dynamically adjusting the search according to the aggregated hints that may be too weak to take effect separately at a single node, we realize high-performance, low-cost search in large-scale unstructured P2P networks.

The contribution of this paper is three-fold.

- We propose a bandwidth-efficient structure (DABF) which can efficiently propagate the hints far away in the network.
- We design a novel DABF-based search mechanism which enables the collaboration of multiple query instances.
- We show through analysis and simulations that we achieve an order of magnitude reduction in search latency over previous proposals.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries. Section 3 presents the design of DABF. Section 4 introduces the DABF-based collaborative search algorithm. The theoretical analysis is presented in Section 5, followed by extensive simulations in Section 6. Section 7 discusses related work and Section 8 concludes the paper.

2. PRELIMINARIES

In this section we first review the concept of basic bloom filters, and then present a brief overview of our DABF-based solution for efficient P2P resource discovery.

2.1. Bloom filters

Bloom filters (BF) [9] provide a technique for set membership lookups. Specifically, given an n-element set $S = \{s_1, s_2, \cdots, s_n\}$, BF uses k independent hash functions, hash_1, hash_2, …, hash_k with the same value range $\{0,1,\cdots,m-1\}$, to describe S by an m-bit vector V. The calculation of V is as follows.

(i) For each element $s_i \in S$ calculate the corresponding m-bit vector V_i. At the beginning the m bits of V_i are all initialized to 0. Then the k hash functions are evaluated over s_i, i.e., $h_1 = \text{hash}_1(s_i)$, $h_2 = \text{hash}_2(s_i)$, …, $h_k = \text{hash}_k(s_i)$. And then the bits in V_i indexed by h_i, $i = 1,2,\ldots,k$, are set to 1, which is denoted as $V_i = \text{BF}(s_i)$.

(ii) Calculate the bitwise-or of all V_i: $V = V_1 \mid V_2 \mid \ldots \mid V_n$.

Let $n = 2$, $S = \{X, Y\}$, $m = 10$, $k = 3$. Fig. 1 illustrates the calculation of the BF vector V of S.

Figure 1. Example of the bloom filter calculation.

To check whether an element x belongs to S, BF looks at the bits in V indexed by $\text{hash}_1(x)$, $\text{hash}_2(x)$, …, $\text{hash}_k(x)$, and considers x in S if and only if all these bits are 1.

By this means BF can "economically" record which elements belong to a given set S. However, there is a certain rate that the judgment is wrong, which is called *false positive* [9]. That is, for an element $s' \notin S$, BF might return a wrong judgment that s' belongs to S. This is because all the bits in V indexed by $\text{hash}_1(s')$, $\text{hash}_2(s')$, …, $\text{hash}_k(s')$ could be set by other elements. This error rate, denoted as P_{err}, is given by:

$$P_{err} = \left(1 - \left(1 - 1/m\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k. \quad (1)$$

where n, k, m are the number of elements in the set, the number of hash functions, and the number of bits in a vector, respectively.

2.2. Overview: DABF-based hint dissemination and aggregation

Current BF-based schemes suffer from a high possibility of false positive due to the large number of resources distributed in the network. To address this problem, the main idea of our proposal is as follows: divide the hint information and disseminate the fragments in a bandwidth-efficient manner (described later); and aggregate the fragment information and make intelligent search decisions through collaboration among multiple instances of a query.

Figure 2. DABF-based hint dissemination and aggregation.

Fig. 2 illustrates an example of collaborative search. Let node A be the query originator for resource X, and node H the resource host. Suppose that the hint information is disseminated at most to nodes E, F and G. Intuitively, the region near H must have stronger information than faraway regions (since during information dissemination only partial information propagated after each hop). So, when the query messages reach the informed nodes E, F and G, node A can aggregate all the returned hints and decide which region is more likely to reach

resource X, even if there is noise from node I. Then node A can rapidly and efficiently find resource X by adding more query messages in the promising region and weakening or even terminating the search in others.

3. DISTANCE-AWARE BLOOM FILTERS

The traditional BF technique, as well as its variations, propagates hint information to all the direct neighbors indiscriminately. However, nodes in large-scale networks might be quite different in many aspects, especially, the bandwidth.

This section introduces DABF, which divides a BF vector into subvectors to disseminate the fragments in a per-neighbor bandwidth-aware manner, i.e., the subvector sent to a neighbor is proportional to that neighbor's available bandwidth, letting a capable neighbor receive a bigger chunk of the vector. In order to explicitly inform how far the hint originator, i.e., the resource, might be, DABF uses a two-dimensional BF set where the column number denotes how many hops the hint has been propagated through.

3.1. Two-dimensional BF set

Since hints are divided and disseminated according to the neighbors' bandwidth, the information strength is not always negatively correlated to the propagated distance. Thus we cannot judge the distance to target resources only by the information strength detected during the search. To address this problem, this subsection presents the structure of a two-dimensional BF set. Node A maintains for each of its neighbors ξ BF vectors. The value of ξ can be predefined by each node according to its link capacity, i.e., the more capacity results in the greater ξ. For a neighbor B of node A:

- The ith BF vector ($1 \le i < \xi$) represents the hints received from B and originated by nodes i hops away from A;
- The ξth BF vector represents the hints received from B and originated by nodes at least ξ hops away from A.

Besides the BF set, each node also maintains a traditional BF vector to represent its local resources.

Each node publishes the hint information of its local resources, and propagates and stores the hints received from its neighbors, which will be introduced in the next subsection. During the propagation of hints, a node stores the received vector in the corresponding element in its BF set according to (i) the neighbor from

which the hint comes to decide the row number, and (ii) the number of hops that the hint has been propagated through to decide the column number.

The main advantage of the two-dimensional BF set is that it explicitly denotes the distance of the hint originator by the column number and thus reducing the false positive possibility, compared with traditional BF techniques that have no distance information. See Section 5 for details. For the sake of convenience, in the rest of this paper we will use the following symbols:

- *UPDATE(A,B,j)*: the update message that node *A* propagates to its neighbor *B*. This message has passed *j* hops when it arrives at node *A* and *j* = 0 means that *A* is the originator of this message.
- *Mask*(*A*,*B*): a binary mask that is used by node *A* to generate the subvector for its neighbor *B*.
- <*Mask*(*A*,*B*)>: the number of non-zero bits in *Mask*(*A*,*B*).
- BL_A: the bandwidth limitation of link capacity of node *A*.

Note that similar to most relevant studies [1],[6], this paper assumes that bandwidth is the main factor that influences the propagation of hints. However, by simply changing BL_A into other attributes (such as computing or storage capacity), the proposed DABF structure can also be applied to the corresponding limitations.

3.2. DABF propagation

Based on the fact that the bandwidth consumption is determined NOT by the storage of BF vectors, but by the amount of propagated **delta information**, we propose to divide and disseminate the hint information to each neighbor proportional to the neighbor's bandwidth. For the sake of convenience, we will use an example topology shown in Fig. 2 to introduce our solution.

Let the number of bits in the BF vector be *m*. And suppose that some resource is *added* to or *deleted* from a node *A*. The processing is as follows.

(i) Node *A* checks the BF vector of its local resource set. If there is no change then nothing needs to be propagated.

(ii) Otherwise, node *A* computes the difference between the new BF vector and the old one, by taking bitwise-XOR.

(iii) Node *A* computes s_i, the "fair proportion" for neighbor *i* according to the bandwidth limitation:

$$s_i = \frac{BL_i}{\sum_{neighbors(A)} BL_j}, \quad (2)$$

where *i* = *B*, *C*, *D*, and *neighbors*(*A*) denotes all the neighbors of *A*.

(iv) Node A generates binary masks $Mask(A,i)$ for $i = B, C, D$, satisfying: $Mask(A,i)$ & $Mask(A,j) = 00\ldots0$ for any $i \neq j$; $Mask(A,B) \mid Mask(A,C) \mid Mask(A,D) = 11\ldots1$; and $<Mask(A,i)> / m = s_i$. For example, suppose that $m = 10$, $s_B = 0.2$, $s_C = 0.3$, and $s_D = 0.5$, then the binary masks *might* be: $Mask(A,B) = 0011000000$, $Mask(A,C) = 1000011000$, and $Mask(A,D) = 0100100111$.

(v) Node A constructs $UPDATE(A,i,0)$, the update message for i, by taking the bitwise-AND between $Mask(A,i)$ and $\delta(BF(A))$, preserving only proportion s_i of A's delta BF vector. Then node A propagates $UPDATE(A,i,0)$ to i.

(vi) After receiving the update message, node i updates the first vector from A in its BF set by taking bitwise-XOR of the old vector and $UPDATE(A,i,0)$.

In order to reduce the bandwidth consumption, the actual propagation uses a periodic cumulative update mechanism. In Fig. 2, for example, to update the second vector from A of node D, node A first masks the received hints during the period ($Mask(A,D)$ & $UPDATE(i,A,0)$, $i = B,C$), then constructs $UPDATE(A,D,1)$ by taking their union (bitwise-OR) and sends $UPDATE(A,D,1)$ to D. Note that when computing s_D by (2) in this case, $neighbors(A)$ denotes all A's neighbors except the one from which $UPDATE(i,A,0)$ comes.

When a new node (for example, node E) joins in the network and becomes a neighbor of node D, it can be regarded as a special case of the above update process, in which the difference equals to the BF vector of node E's local resources set. And when a node decides not to share a resource any longer or it leaves the network, the processing can be viewed as a counterpart of the node join case, which is omitted here due to lack of space.

From the above process, we can see that there are two kinds of information represented by an element in the BF set, namely, the hint strength (number of non-zero bits) and the hint originator distance (column number). If a node does not receive updates from a neighbor for a period, it would consider that neighbor fails and invalid corresponding elements in its BF set.

4. DABF-BASED COLLABORATIVE P2P SEARCH

In this section we first extend the concept of traditional "query messages" to "intelligent units", and then presents the details of the collaborative search algorithm.

4.1. Units

Query messages in most P2P networks can only be passively forwarded (with the only exception that the messages periodically "ask" the query originator whether the target is found and they stop if the answer is yes). EDBF [6] and PST [24] use decaying bloom filters to realize hint information propagation. As shown in Fig.3, in EDBF a node decays the received BF vector by a predefined, unified decay factor before propagating to neighbors. During the search, the query is forwarded to the neighbor having the maximum amount of information. However, according to ref. [6], the informed neighborhood in this method is small, i.e., the hint information out of the informed neighborhood would be too weak to be distinguishable from random noise.

Figure 3. EDBF hint dissemination.

To address the problems of EDBF and PST, we extend the concept of the passive messages and propose active "units". In the rest of this subsection we will explain the improvement compared to traditional routing massages.

Suppose that a unit (say, *M*) arrives at an intermediate node (say, node *A*). Then the active behaviors of *M* include:

(i) *Suspend*, *resume* and *terminate*: We say that *M* is *suspended* at node *A* if *A* temporarily holds *M* without forwarding; that *M resumes* its search if node *A* continues the forwarding; and that *M terminates* if node *A* discards *M*.

(ii) *Move*, *reside* and *dispatch*: We say that *M moves* from *A* to *B* and *M* is a *moving* unit if node *A* forwards *M* to a neighbor *B*; and that *M resides* at *A* and *dispatches* sub units and *M* is a *residing* unit if node *A* forwards *M* to multiple neighbors. The use of *residing* unit is to aggregate the hints obtained by its sub units.

(iii) *Notify* and *order*: A moving unit periodically *notifies* its parent of its obtained hints, and a residing unit periodically *notifies* its parent as well as sub units of its aggregated hints. Note that the *notify* behavior needs the moving unit to carry address information (e.g., the IP address) of its parent. A residing unit may *order* its sub units to execute behaviors like *suspend*, *resume* and *terminate*. The *order* behavior can be implemented through acknowledgements when sub units *notify* their parents. As introduced before, in the publishing phase the hints are divided into subvectors and disseminated to different neighbors, making only partial information available at each node. So, when a unit arrives at an intermediate node *A* and asks whether a neighbor *N* should be taken as the next hop to reach the target resource *x*, we cannot simply answer "yes" or "no", but can only return a value representing the possibility.

Let the BF vector of resource x be U and let the jth vector from neighbor N of node A be V. The *similarity* between resource x and vector V, denoted as $\Omega(x,V)$, is defined as the vector product of x and V (normalized by V) [6]. Clearly, the higher similarity between x and V indicates the higher probability of reaching x through N after j hops.

4.2. Searching algorithm

The basic idea behind our collaborative search algorithm is rather simple: Aggregate the hints obtained by different units and judge which region is more hopeful, then dispatch more units to the promising region to accelerate the search, and suspend or even terminate search in other regions to cut down the message cost.

Let the target resource be x. Suppose that a unit arrives at node A and its parent unit resides at node P, which are referred to as U_A and U_P respectively. Define

$$Max(A) = \text{Max } \Omega \text{ found by } U_A. \quad (3)$$

The *CollaborativeSearch* algorithm at A is as follows.

(i) U_A computes Ω between x and each BF vector in node A's BF set by (3), then concludes the local $Max(A)$ and *notifies* its parent U_P.

(ii) U_A chooses a subset of neighbors $\{N\}$ satisfying

$$\Omega(x,V) \geq \alpha \times Max(P), \quad (4)$$

where V is the jth vector from neighbor N of node A and $\alpha \leq 1$ is a predefined fault-tolerant parameter. U_A *dispatches* a sub unit to each node in the subset. If the subset is null then U_A would *move* to the neighbor with nonzero $Max(A)$. If $Max(A) = 0$ then U_A *moves* to a randomly chosen neighbor.

(iii) U_P aggregates the notifications from all of its sub units as well as parent to update $Max(P)$ and *notifies* the new $Max(P)$ to them including U_A. U_A reexamines the conditions as discussed in the above Step (ii) and chooses the subset again. If U_A has been *suspended* and the new subset is not empty then U_A will *resume* its search by *dispatching* a sub unit to each node in the new subset; and if some node in the old subset is excluded outside the new subset then U_A will *order* the corresponding sub unit to *suspend*.

(iv) As discussed in Section 3, besides the similarity between the elements and the target, the column numbers of elements represent the distance from possible target resources. Let the column number of the element with $Max(A)$ be $j_{\max}$, then U_A *dispatches* a sub unit U_N to each neighbor N satisfying

$$\Omega(x,V) \geq Threshold, \quad (5)$$

where V is the jth vector from neighbor N ($1 \le j < j_{\max}$) and *Threshold* is a predefined parameter.

(v) U_N chooses the maximum value of j satisfying (5) as its *TTL*. U_N is used to check whether x can be found after j hops through neighbor N. If N has at least one neighbor V satisfying

$$\Omega(x,V') \ge \Omega(x,V), \quad (6)$$

where V and V' are respectively the jth and j'th vectors from neighbor N ($j' < j$), then U_N will *move* (or *dispatch* sub units) to node V, and the following processing is similar to the case at node A in Step (i).

(vi) Otherwise it indicates that U_N cannot reach x and thus U_N would *terminate* its search immediately.

In this procedure, the sub units and parent units use *local* aggregation to ease the burden of the query originator, and all the units utilize a periodic and cumulative communication pattern between each other to reduce the overall traffic in the network. Let k be the number of hash functions. Suppose that the degree of node A is d and the number of columns in the BF set is c. Then the average time complexity of this algorithm is $\mathrm{O}\left(k \times (d-1) \times c\right)$.

Let *gMAX* denote the maximum similarity with the target resource that has been obtained by all the units. By (3) it is easy to see that *gMAX* roughly indicates the minimum distance to the target from all units, i.e., a higher *gMAX* means some unit is closer to the target. Similar to traditional BF techniques, however, since the computation of similarity might be affected by noise, there is certain possibility that false positive errors occur. See next section for detailed analysis.

(vii) Clearly, the collaborative search algorithm takes no effect when the possibility of false positives is high. To address this problem, we utilize two similarity thresholds Th_1 and Th_2 ($Th_2 < Th_1$) and divide the search process into three phases.

(vii a) When $gMAX \ge Th_1$ we consider that the hint information is dominant to the noise and the unit obtaining *gMAX* (say, unit M) should be very close to the target resource. So, we can suspend all the units except M to reduce the message cost.

(vii b) Similarly, at the beginning when $gMAX < Th_2$ we consider that the noise is dominant to the hints and all the units are very far from the target resource, which means the *CollaborativeSearch* algorithm is not efficient. For this phase, we propose the following *AccelerateSearch* algorithm. If after consecutive λ hops *gMAX* is still less than Th_2, then each unit obtaining the top-k similarities with the target will *dispatch* γ sub units. λ is referred to as *search acceleration period*.

This process is stopped when *gMAX* is greater than or equal to Th_2 and all the units with similarity less than $\mu \times gMAX$ will be *ordered* to *terminate*.

(vii c) If $Th_2 \leq gMAX < Th_1$, we consider that the hints are competitive with the noise. And the *CollaborativeSearch* algorithm is adopted in this phase.

We will show through evaluations in Section 6.3 that the above mechanism effectively controls the total number of units as well as the message cost for collaborations between units.

5. ANALYSIS

This section first proposes a simplified analytical model, and then analyzes the DABF error rate and CPS routing performance.

In order to handle the uncertainty and complexity of unstructured P2P networks in our analysis, we propose a reasonably simplified analytical model. In the ideal design, at the beginning of the search if after consecutive λ hops *gMAX* is still less than Th_2, then each of the units that obtain the top-*k* similarities will dispatch at most γ sub units. In contrast, in the simplified model each of the top-*k* units dispatches sub units to all the direct neighbors except the one it comes from. We will show the effects of the simplified model by comparing the analytical result with the simulation result in Section 6.

CPS may make wrong judgments when choosing the forwarding subset *S* at an intermediate node *A*. Suppose that neighbor *N* is the correct next hop, i.e., neighbor *N* is on the shortest path to the target resource among all the neighbors. There might be two types of errors when choosing *S*:

- If *S* contains neighbors besides *N*, we refer to it as *type-I error*; and
- If *S* does not contain *N*, we refer to it as *type-II erro*r.

As discussed in the above section, the units in wrong directions can quickly discover the error and terminate their search at the price of a modestly increased number of messages, which will be evaluated in the next section. So the type-I error is subordinate. We will evaluate the total cost of CPS for searching an object (thus including the extra cost of type-I error) through simulations in the next section. On the other hand, the type-II error may result in an obvious increase in search latency, thus it dominates the type-I error.

Note that similar to other unstructured P2P networks [4],[5],[6], by (1) there is always a false positive possibility when deciding whether the target *x* belongs to node *A*. To address this problem, CPS first computes the similarity by (3), and if it

equals to 1 then the query originator would directly contacts with A to make the final decision. This case is simple and thus being omitted here.

In the rest of this paper we will use the (type-II) *error rate* to represent the efficiency of CPS routing, the definition of which is as follows.

Definition 1. *At an intermediate node A that is i hops away from the target, the* ***error rate*** *is defined as the possibility that a type-II error occurs.*

If a type-II error occurs it is referred to as failed routing, otherwise it is referred to as *effective routing*.

Given a neighbor N of node A, we use p to denote N's link bandwidth proportion among the neighbors of A, i.e.,

$$p = \frac{BL_N}{\sum_{neighbors(A)} BL_i}, \quad (7)$$

where *neighbors*(A) denotes A's neighbors which receive the BF vector.

Clearly the same target x has the same denominator, so we can use only the numerator to represent the similarity with x of each element in the BF set. This subsection will analyze the influence of noise on the jth BF vector V from neighbor N of an intermediate node A, i.e., the number of non-zero bits that are indexed by the k hash functions in V due to noise.

For any node U, the expected number of non-zero bits in the BF vector of local resources hosted by U, denoted as w, is given by:

$$w = (1-(1-\frac{1}{m})^{kg}) \times m \approx (1-e^{-\frac{kg}{m}}) \times m . \quad (8)$$

Let S_i denote the expectation of the total number of non-zero bits which are caused by the resources hosted by nodes i hops away from A. Since there are ξ BF vectors in each column, each BF vector in the ith column has S_i / ξ uniformly random non-zero bits. We use the random variable Y_i to denote the number of non-zero bits that are indexed by the k hash functions in V due to noise. Let $\mathrm{P}(y)$ denote the probability that the value of Y_i is y, where y is a non-negative integer no greater than k. Then

$$\mathrm{P}(y) = C_k^y \frac{(kgq^{i-1})^y (m - kgq^{i-1})^{k-y}}{m^k} . \quad (9)$$

Suppose that the hints of target resource reach node A through its neighbor B after h hops propagation. If $h < \xi$ the information is stored in the hth vector (V) from neighbor B of A's BF set; otherwise it is stored in the ξth vector from B.

The probability that a bit indexed by k hash functions in V is non-zero, denoted as $r(h)$, is composed of two parts.

The first comes from the target resource h hops away with the probability:

$$\rho = \prod_{j=1}^{h-1} p_j \ . \quad (10)$$

And the second comes from the noise with the probability kgq^{i-1}/m. We use the random variable X to denote the number of non-zero bits that are indexed by the k hash functions in V ($h < \xi$) or V' ($h \geq \xi$). Let $\mathrm{P}(x)$ denote the probability that the value of X is x, where x is a non-negative integer no greater than k. Then we have

$$\mathrm{P}(x) = C_k^x \frac{(1+\rho)^{k-x}(m-kgq^{h-1})^{k-x}((1+\rho)kgq^{h-1}-\rho m)^x}{m^x} \ . \quad (11)$$

During the routing, suppose that a unit arrives at node A and its parent unit resides at node P. Let W' be the h'th vector from neighbor B of A ($h'=h$ if $h<\xi$, otherwise $h'=\xi$). Then according to Step (ii) in the collaborative search algorithm, a sub unit would be dispatched to a correct neighbor N, i.e., it would be an effective routing, if and only if

$$\forall N \neq B, \forall i \in [1, \xi], \Omega(x, W) \geq \mu\Omega(x, V') \ . \quad (12)$$

where V' is the ith vector from neighbor N of A.

Let τ denote the probability of the node degree is d and suppose node degrees uniformly distributed in the interval [a, b]. Then, the routing error rate $\Psi(h)$ at an intermediate node h hops away from the target resource are given by:

$$\Psi(h) = \tau \sum_{d=a}^{b} (1 - (\prod_{i=1}^{(a+b)/2} \mathrm{P}(X \geq \mu Y_i))^{d-1})(1 - 1/d) \ . \quad (13)$$

Let x be the distance between the query originator and the target resource. If the query originator is very close to the target resource, i.e., $x \leq \alpha$, we can approximately consider each routing is effective. If the distance satisfies $\alpha < x \leq \beta$, then the hints are competitive with the noise. If the query originator is very far away from the target resource ($x > \beta$), then at the beginning of the search we will utilize the accelerating algorithm with the simplification that all the moving units (top 100%) dispatches sub units to their neighbors in each acceleration period (λ). Let λ be the number of consecutive hops between two acceleration in the *AccelerateSearch* algorithm.

Suppose that there are $\phi(x)$ nodes that are h hops away from the target resource. Suppose that the maximum distance between the target resource and any node is χ and there are n nodes in the network. The number of nodes that are χ hops away from the target resource, denoted as $\phi(\chi)$, is given by:

$$\phi(\chi) \approx \frac{(a+b)(n-(a+b-2)^{\chi-1}+2^{\chi-1})-4n}{a+b-4}. \quad (14)$$

The average search latency (D) is given by:

$$D=\frac{1}{n}(\sum_{x=1}^{\alpha}\frac{x(a+b)(a+b-2)^{x-1}}{2^x}+$$
$$\sum_{x=\alpha+1}^{\beta}\frac{(a+b)(a+b-2)^{x-1}}{2^x}\sum_{i=\alpha+1}^{x}\tau\sum_{d=a}^{b}(1-(\prod_{i=1}^{(a+b)/2}\mathrm{P}(X\geq\alpha Y_i))^{d-1})(1-1/d)+ \quad (15)$$
$$\sum_{x=\beta+1}^{\chi}\frac{(a+b)(a+b-2)^{x-1}}{2^x}(\lambda(x-1)+\beta+\frac{a+b-2}{2}\sum_{i=\alpha+1}^{\beta}\tau\sum_{d=a}^{b}(1-(\prod_{i=1}^{(a+b)/2}\mathrm{P}(X\geq\alpha Y_i))^{d-1})(1-1/d)))$$

6. EVALUATION

This section evaluates different aspects of the DABF-based CPS algorithm, and compares CPS with other P2P search schemes including Gnutella (flooding) and SQR [6].

- Section 6.1 introduces basic settings in our experiments;
- Section 6.2 evaluates the search latency of different schemes and shows that the performance of CPS is close to flooding;
- Section 6.3 evaluates the message cost during search of different schemes and shows that CPS achieves a relatively low cost;
- Section 6.4 evaluates CPS with different parameter combinations;
- Section 6.5 evaluates the performance of CPS under churn and shows that CPS is robust even if there are a large number of nodes fail; and
- Section 6.6 evaluates the impact of replications on the performance of CPS.

All the evaluations are conducted on a workstation with an Intel Core 2 Duo 3GHz CPU and 3GB memory. And the experiment for each property is repeated at least 1,000 times.

6.1. Methodology

For all search schemes, the network size (n) is varied from 2,500 to 80,000. The node degree is uniformly distributed with the mean value (c) of 4. There are 10,000 different resources in the network and on average each node has 2 resources (g). The distributions of resources and searches are both uniform. The number of hash functions (k) is 64.

For CPS we ran experiments with the following settings. The width of DABF vectors (m) is set to 64K and 64K×5 bits, respectively. The second DABF width is to be compared with SQR (see the following paragraph). The random variable $\log p$ is uniformly distributed with the mean value of $\log 0.3$, where p denotes the proportion of node bandwidth limitation (see Definition 1). The fault-tolerant factor (α) is 0.5. The number of units at the beginning is 3. The number of hops between two search acceleration (λ) is 3, and in an acceleration each of the top 50% units *dispatches* $\gamma = 3$ sub units to the neighbors. The column number (c) of the DABF sets is 5. And the two thresholds for collaborative search and accelerating search (Th_1 and Th_2) are set to 1/4 and 1/16, respectively.

In SQR [6] a node maintains a BF vector for each of its neighbors. It decays received BF vectors by a predefined and unified decay factor (f) and propagates to others, i.e., the proportion of propagated hints is f. During the search, query messages are routed to neighbors having the maximum amount of hints. SQR is efficient when the network size is small. However, since its decayed hints can propagate only in a small neighborhood, SQR has a poor performance in large-scale networks. For example, in our simulations when the network size is 2,500, the search latency of SQR is about 130 hops, more than an order of magnitude higher than others. So, we equip SQR with an enhancement similar to the accelerating search algorithm. That is, we check the similarity between the target and the hints every 3 hops, and if it is less than a threshold (Th_2) we would forward the query to all direct neighbors. We refer to the SQR scheme with the acceleration mechanism as SQR-A. For SQR-A, the width of BF vectors (m) is set to 64K×5 bits to be competitive with CPS's 5-column BF set 64K-bit width. Since SQR cannot distinguish nodes with different bandwidth limitations, we simply set the decay factor to the same value of p_0, i.e., $f = 0.3$.

For Gnutella we use no hint information and simply flood query messages during the search.

The parameter settings are summarized in Table 1.

TABLE 1. PARAMETERS OF EXPERIMENTS

6.2. Search Latency

We evaluate the average search latency (measured in terms of overlay hops) of CPS, as a function of the number of nodes. In each experiment we randomly

choose a query originator and a target resource, and the query messages are routed using the proposed CPS search algorithm.

The results are plotted in Fig. 4. We compare CPS and CPS (64K*5) with other P2P search schemes including SQR-A and Gnutella (Gnu). The analytical results of CPS (CPS-T) are also included in this figure for comparison. Our conclusion for this figure is three-fold.

Figure 4. Search latency of CPS, SQR and Gnutella.

First, the search latency of CPS is close to that of Gnutella. Note that Gnutella uses simple flooding mechanisms without any message control, meaning that the latency of Gnutella is the lower bound for all schemes. So, we conclude that CPS almost achieves the lowest search latency.

Second, the latency of CPS is obviously lower than that of SQR-A. For example, when the number of nodes is 2,500, CPS has about 43% reduction in latency over SQR-A and CPS (64K*5) has more reduction. This is mainly because in contrast to SQR-A the BF set in CPS contains information about the distance to the target, and the aggregation of incomplete hints in collaborative search is stronger than separate hints scattered in the network.

Third, our analytical results are close to the experimental search latency. For example, when the number of nodes is 2,500, the difference is less than 31% of the analytical latency. The main reason for this difference is that we use a relatively imprecise topology model in our analysis, while other simplifications discussed in Section 5 also take subordinate effects.

6.3. Message cost

We evaluate the average message cost (including the cost caused by the collaborations between units) of CPS, measured in terms of the number of messages propagated during the search, as a function of the number of nodes. Similar to the above experiment, we randomly choose a query originator and a target resource to conduct a query. We also evaluate the message cost of Gnutella and SQR-A for comparison.

Figure 5. Message cost of CPS, SQR and Gnutella.

The results are shown in Fig. 5. From this figure we can see that the message cost of CPS is much lower than that of SQR-A. This is because (i) CPS propagates the hints to more nodes than SQR-A; and (ii) the collaborative search

algorithm controls the search in a more intelligent manner, which consequently saves much unnecessary message cost. On the other hand, we cannot see the message cost of Gnutella in Fig. 5. This is because it is too high to display in this figure (more than 4,000 messages when n=2,500, and more than 110,000 messages when n=80,000).

6.4. Impact of different parameters

This subsection evaluates the average search latency and message cost with different parameter settings, including the proportion of node bandwidth limitation (p) and the search acceleration period (λ), for a fixed network size of 2,500 nodes. Other parameters are the same as introduced in Section 6.

Figure 6. Impact of different parameter settings on the latency and cost of CPS.

The results are plotted in Fig. 6. We use CPS(p, λ) to denote CPS with proportion p and period λ. From this figure we can see that (i) as p increases, both the search latency and the message cost decreases; and (ii) as λ increases, the search latency increases modestly while the message cost decreases. Based on the above observation, we can achieve an appropriate tradeoff between the search latency and message cost by choosing different combinations of p and λ.

6.5. Robustness

In large-scale networks, nodes might join/leave the network frequently, inducing an unstable state called churn [10]. A query might be improperly responded due to the failure of intermediate nodes. This subsection evaluates the robustness of CPS under churn.

In our experiment, we simulate network churn by choosing faulty nodes randomly and uniformly from the network of 2,500 nodes and assuming the hint propagation still has not be finished, i.e., the units might obtain wrong hints. The percentage of faulty nodes is varied from 0 to 20%, and in order to highlight the robustness of CPS we assure that the network is always connected. Obviously, without any limitation all queries can be answered with some certain cost. So, we consider a query to be failed when the cost exceeds 1,000 messages.

Fig. 7 plots the percentage of failed queries of CPS. This is computed by simulating 10,000 queries targeted to the existing resources that are randomly chosen from the resource pool.

Figure 7. Robustness of CPS.

From this figure we can see that most (more than 98%) queries can successfully reach the target resources. This is mainly because the fault-tolerant factor (α) assures to *dispatch* sub units to other neighbors besides the correct but faulty ones. Therefore, we conclude that our CPS scheme is robust even if there are a large number of nodes fail.

6.6. Impact of replication

Replication of the same resource at different nodes improves the efficiency of search [6]. In our experiment we use the Zipf distribution for replication which is frequently used to model the replication on the Internet. In the Zipf distribution, the *i*th most replicated resource has 1/*i* times as many replicas as the most replicated resource. We use a replication rate of 10% for the most replicated resource. The network size is fixed to 2,500 nodes and other parameters are the same as introduced in Section 6.

Figure 8. Impact of replication on the latency and cost of CPS.

The results are shown in Fig. 8, where we use CPS-R to represent CPS with the Zipf replication. From this figure we can see that (i) the average latency (measured in terms of the number of hops) of CPS-R is about 87.2% of the standard CPS; and (ii) the average cost (measured in terms of the number of messages) of CPS-R is only about 80.5% of the standard CPS. This experiment proves that CPS benefits a lot from the resource replication mechanism.

7. RELATED WORK

Unstructured P2P networks have a loosely coupled topology structure that is easy to maintain. So, different from structured P2P networks, the hot-spot for unstructured P2P networks is how to realize efficient search without topology knowledge [35].

Gnutella [2] uses constrained flooding to realize search. Its variation, mainly including iterative deepening [11], in turn starts multiple scoped breadth-first searches (BFS) until all queries are satisfied. Gnutella and its variations have simple maintenance and good fault-tolerance characteristics. They also support

flexible search conditions [12]. However, they consume much bandwidth, which greatly limits their scalability.

To address this problem, Random Walk (RW) [3] is proposed with the idea of forwarding queries to a randomly chosen neighbor. Its variations mainly include BRW [1], ARW [4], *k*-RW [5], and RBFS [13]. Biased Random Walk (BRW) [1] chooses the most powerful neighbor for query forwarding and caching pointers to content located one hop away. Accelerated Random Walk (ARW) [4] adds walker self-replication feature to Random Walk, and replicates the query messages *k* copies every *m* hops. *k*-way Random Walk (*k*-RW) [5] randomly chooses *k* neighbors to forward queries. RBFS [13] uses a breadth-first search manner to randomly forward queries to neighbors.

All the above schemes can be classified into the "blind" search category, which is blind to the hints information during the search. These schemes produce large amounts of redundant messages and consume much bandwidth.

This problem has motivated a lot of studies about the "informed" search schemes, which use hint information to guide queries. Most informed schemes propagate hints based on bloom filters (BF) [9], which provide an efficient approach to set membership lookups in unstructured P2P networks.

PlanetP [7] utilizes BF vectors to represent the keywords of the shared files and propagates the vectors using gossiping. Attenuated search [8] uses a collection of *d* BF vectors with the *i*th vector being associated with content published by nodes *i* hops away. These proposals can propagate resource information only one or two hops away due to the challenge of maintenance cost and false positive possibility.

Dynamic bloom filter (DBF) [15] replaces traditional BF vectors with a group of vectors and nodes can adjust the number of used BF vectors according to their actual needs, which efficiently lowers the false positive possibility. Ref. [23] proposes a similar approach to using BF vectors to propagate information. Obviously, these proposals are orthogonal to our DABF, and in our future work we will add the dynamic features to CPS to achieve higher scalability.

SQR [6] uses exponentially decaying bloom filter (EDBF) to realize hint information propagation. Analysis and simulations show that SQR is effective in small-scale networks. However, according to ref. [6], the informed neighborhood in SQR is very small. Therefore SQR takes almost no effect in large-scale (e.g., 10,000) networks. Moreover, in realistic P2P networks different nodes usually have different bandwidth limitation. For SQR, it is straightforward that low-bandwidth neighbors should have higher decay factors to reduce the message cost of hint updates. However, the predefined, unified decay factor of EDBF cannot adapt to the heterogeneity of realistic P2P networks.

To address this problem, PST [24] proposes to support custom discarding rates for each neighbor. But its randomized discarding method is likely to discard all information at the first few hops in the propagation process, making it have a poor searching performance in large networks with more than tens of thousands of nodes. SQR and PST can be viewed as the preliminary form of DABF. The key differences between them include: (i) DABF uses deterministic masks (instead of random discarding) for information propagation; and (ii) DABF designs intelligent units, which support a series of active behaviors such as suspend, resume, terminate, move, reside, dispatch, notify and order.

The bloom filter technique has also been widely used in many other fields. For example, Approximate Concurrent State Machines (ACSM) [16] enables the BF technique to support network state lookups by improving the following aspects: (i) the false positive problem, (ii) associating value to BF, (iii) being easy to delete; and 4) adding notion of time. Ref. [17] discusses the false negative problem of counting bloom filters. Refs. [18] and [19] respectively discusses how to utilize bloom filters to realize hint-based WSN routing and data-centric routing. Space precludes further discussion.

In order to get better tradeoff between the search latency and message cost, BubbleStorm [14] proposes a simple probabilistic search mechanism without using BF. It is built on random multigraphs and has a flexible and reliable strategy for performing exhaustive search by exploiting the heterogeneous bandwidth of nodes. The provided search guarantees are tunable, with success probability adjustable well into the realm of reliable systems. Clearly, the topology of BubbleStorm can be viewed as a half-structured network.

An important topic for P2P search is the complex search in P2P networks, which pays more attention on the organization of distributed data indices. For example, Astrolabe [20] develops a P2P information management system that supports aggregate query on large-scale system state, permitting rapid updates and providing on-the-fly attribute aggregation. Astrolabe uses a restricted form of mobile code based on the SQL query language for aggregation, and gives rise to a novel consistency model. Ref. [21] proposes a skyline-based P2P search scheme for efficient top-k query processing. And ref. [22] discusses how to accelerate complex P2P search using peer caching.

Ref. [25] considers information aggregation as a method for reducing the information exchanged in a grid network and used by the resource manager in order to make scheduling decisions. Information is summarized across nodes and sensitive or detailed information can be kept private, while resources are still publicly available for use. The authors also present a general framework for information aggregation trying to identify issues that relate to aggregation, and describe a number of techniques including single point and intra-domain

aggregation, which define appropriate grid-specific domination relations and operators for aggregating static and dynamic resource information.

Ref. [26] presents a decentralized market-based approach to resource discovery in a heterogeneous overlay network. This strategy dynamically assigns resources in an overlay network to requests for service based on current system utilization, thus enabling the system to accommodate fluctuating demand for its resources. The authors treat the allocation of system resources as a constrained optimization problem and propose a mathematical model and derive a simple but efficient decentralized algorithm.

Ref. [27] proposes Cobweb clustering based network-aware summarisation algorithms for the resource discovery in P2P-content networks. The authors use precision–recall based metrics to compare the accuracy for specific types of queries generated over the summarised content, and identify how summarisation can improve the discovery process while also improving the accuracy of the discovered resource(s).

In order to reduce the number of message passes needed to achieve the efficient resource management, Ref. [28] proposed an interconnection structure called the Distributed Spanning Tree (DST). The application of DST converts the peer network into logical layered structures and thereby provides a hierarchical mechanism for resource management.

Ref. [29] presents a strategy and a system, called ERGOT, that combine DHTs and SONs to enable semantic-based service discovery in distributed infrastructures such as Grids and Clouds. ERGOT uses semantic annotations that enrich service specifications in two ways: (i) services are advertised in the DHT on the basis of their annotations, thus allowing us to establish a SON between service providers; (ii) annotations enable semantic-based service matchmaking, using a similarity measure between service requests and descriptions.

Ref. [30] presents a proactive data replication mechanism for P2P based Video-on-Demand platform, where a peer can proactively replicate data chunks to stable cache servers for future sharing, when it has high possibility to leave the overlay. The authors propose two key heuristic algorithms for departure prediction and replicating chunks selection, and design the cache servers management mechanism.

In order to maximize the QoS, Ref [31] proposes a capacity and load-aware service selection (CLSS) and peer cache-based service discovery (PCSD) algorithms to maximize profit with minimum resource consumption in P2P overlays. The authors use a heuristic method and the problem of service discovery and service selection is formulated as a linear programming (LP) problem together with the constraints and proposed algorithms.

Ref. [32] proposes a weighted tree based algorithm for resource discovery, using a bitmap where the number of bit positions is proportional to the number of attributes for each resource. In every node of the tree there will be a footprint of existence resources in children and the descendant of that node, and when the user's request reaches each node, if the requested resources are available in its children or the descendant, it can directly and without referring to extra nodes and unnecessary traffic reach the node which owns the required resource.

Ref. [33] proposes a distributed and hierarchical mechanism for improving the speed of resource discovery, where a root layer is added as coverage for the service index of under layers and nodes in the same level (which have the same parent) send queries to each together directly.

Ref. [34] proposes a scalable DHT- and ontology-based Information Service (DIS), which organizes resources into a DHT ring. Only stable virtual organizations can join DIS via a new DHT node, whereas volatile organizations join DIS via being the sub-domain of other organizations. Ontology-based information integration is adopted in DIS, which supports semantic-based information queries. By integrating DHT- and semantic-based query techniques, DIS speeds up the information query and improves the query precision and integrity.

8. CONCLUSION

Bloom filter-based P2P search schemes in large-scale networks suffer from a high possibility of false positives. To address this problem, in this paper we present the DABF structure and design a DABF-based collaborative P2P search algorithm.

Currently our proposals do not take into account for the adaptation to different churn rates of P2P systems, which is an important issue that merits further exploration. Furthermore, the methods of deciding the link bandwidth for DABF is still open and reserved to our further work. On the other hand, this paper focuses on exact-match P2P search, while currently more complicated search, e.g., range, top-*k* and skyline queries, is becoming increasingly popular. Therefore, it would be interesting to enable DABF to support such complicated search schemes.

ACKNOWLEDGMENT

This paper is sponsored in part by the National Natural Science Foundation of China (NFSC) under Grant No. 60903205 and 61222205, the National Basic Research Program of China (973) under Grant No 2009CB320503, and the Research Fund for the Doctoral Program of Higher Education (RFDP) under Grant No. 20094307110008. The second author acknowledges the partial support by grants from NSF NetSE and NSF CyberTrust program, an IBM faculty award and an Intel ISTC on Cloud Computing.

REFERENCES

1. Carlini E., Coppola M., Dazzi P., Laforenza D., Martinelli S., Ricci L.. Service and Resource Discovery supports over P2P overlays. ICUMT 2009: 1-8
2. Baraglia R., Dazzi P., Mordacchini M., Perego R., Ricci L.. Gossip Communities: Collaborative Filtering Through Peer-to-Peer Overlays. SEBD 2010: 54-61
3. Gkantsidis C, Mihail M, Saberi A. Random Walks in Peer-to-Peer Networks. *Proc. IEEE Infocom*, 2004.
4. Zheng Q, X Lu, Peng W. An Efficient Random Walks Based Approach to Reducing File Locating Delay in Unstructured P2P Network. *Proc. GLOBECOM*, 2005.
5. Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and Replication in Unstructured Peer-to-Peer Networks. *Proc. 16th ACM International Conference on Supercomputing (ICS)*, New York, USA, June 2002.
6. Kumar A, Xu J, Zegura EW. Efficient and Scalable Query Routing for Unstructured Peer-to-Peer Networks. *Proc. IEEE Infocom*, 2005.
7. Francisco M, Christopher P, Richard P, Thu D. PlanetP: Using Gossiping to Build Content Addressable P2P Information Sharing Communities. *Technical Report, DCS-TR-487*, Piscataway, Rutgers University, 2002.
8. Rhea SC, Kubiatowicz J. Probabilistic Location and Routing. *Proc. IEEE Infocom*, Mar. 2002.
9. Bloom B. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, vol. 13, no. 7, July 1970.
10. Godfrey PB, Shenker S, Stoica I. Minimizing Churn in Distributed Systems. *Proc. ACM SIGCOMM*, 2006.
11. Yang B, Garcia-Molina H. Efficient Search in Peer-to-Peer Networks. *Proc. 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, July 2002.

12. Ripeanu M, Foster I, Iamnitchi A. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing*, vol. 6, no. 1, January-February 2002.
13. Kalogeraki V, Gunopulos D, Zeinalipour-yazt D. A Local Search Mechanism for Peer-to-Peer Networks. *Proc. 11th ACM Conference on Information and Knowledge Management (CIKM)*, McLean, Washington, USA, 2002.
14. Terpstra WW, Kangasharju J, Leng C, Buchmann AP. BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search. *Proc. SIGCOMM*, 2007.
15. Guo D, Wu J, Chen H, Yuan Y, and Luo X. The Dynamic Bloom Filters. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2010, vol. 22, no. 1, pp. 120-133.
16. Bonomi F, Mitzenmacher M, Panigraphy R, Singh S, Varghese G. Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines. *Proc. SIGCOMM*, 2006.
17. Guo D, Liu Y, Li X, Yang P. False Negative Problem of Counting Bloom Filter. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2010, vol. 22, no. 5, pp. 651-664.
18. Li X, Wu J, Xu J. Hint-based Routing in WSNs Using Scope Decay Bloom Filters. *Proc. International IEEE Workshop on Networking, Architecture, and Storages (NAS)*, 2006. 1-3 Aug. 2006. ISBN: 0-7695-2651-9, DOI: 10.1109/IWNAS.2006.30.
19. Guo D, He Y, and Yang P. Receiver-Oriented Design of Bloom Filters for Data-Centric Routing. *Elsevier Computer Networks Journal (COMNET)*, 2010, vol. 54, no. 1, pp. 165-174.
20. VanRenesse R, Birman KP, Vogels W. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *Proc. TOCS*, 2003.
21. Vlachou A, Doulkeridis C, Noervaag K, Vazirgiannis M. Skyline-based Peer-to-Peer Top-k Query Processing. *Proc. IEEE International Conference on Data Engineering (ICDE)*, 2008.
22. Deb S, Linga P, Rastogi R, Srinivasan A. Accelerating Lookups in P2P Systems Using Peer Caching. *Proc. IEEE International Conference on Data Engineering (ICDE)*, 2008.
23. Koloniari G, Pitoura E. Content-based routing of path queries in peer-to-peer systems. *Proc. EDBT*, 2004.
24. Zhang Y., Lu X. Zheng Q. An Efficient Search Algorithm for Large-Scale P2P Systems. *Journal of Software* (In Chinese), 2008, vol. 19, no. 6, pp. 1473-1480.

25. Panagiotis C. Kokkinos, Emmanouel A. Varvarigos. Scheduling efficiency of resource information aggregation in grid networks. Future Generation Computer Systems, 2012, vol. 28, no. 1, pp. 9-23.
26. Jay Smith, Edwin K. P. Chong, Anthony A. Maciejewski, Howard Jay Siegel. Overlay network resource allocation using a decentralized market-based approach.. Future Generation Computer Systems, 2012, vol. 28, no. 1, pp. 24-35.
27. René Brunner, Agustín C. Caminero, Omer F. Rana, Felix Freitag, Leandro Navarro. Network-aware summarisation for resource discovery in P2P-content networks. Future Generation Computer Systems, 2012, vol. 28, no. 3, pp. 563-572.
28. P. Victer Paul, N. Saravanan, S. K. V. Jayakumar, P. Dhavachelvan, R. Baskaran. QoS enhancements for global replication management in peer to peer networks. Future Generation Computer Systems, 2012, vol. 28, no. 3, pp. 573-582.
29. Giuseppe Pirrò, Domenico Talia, Paolo Trunfio. A DHT-based semantic overlay network for service discovery. Future Generation Computer Systems, 2012, vol. 28, no. 4, pp. 689-707
30. Xiaofei Liao, Hai Jin, Linchen Yu: A novel data replication mechanism in P2P VoD system. Future Generation Computer Systems, 2012, vol. 28, no. 6, pp. 930-939
31. Neeraj Kumar, Rahat Iqbal, Naveen Chilamkurti: Capacity and load-aware service discovery with service selection in peer-to-peer grids. Future Generation Computer Systems, 2012, vol. 28, no. 7, pp. 1090-1099
32. Leyli Mohammad Khanli, Saeed Kargar. FRDT: Footprint Resource Discovery Tree for grids. Future Generation Computer Systems, 2011, vol. 27, no. 2, pp. 148-156
33. Saeed Ebadi, Leyli Mohammad Khanli. A new distributed and hierarchical mechanism for service discovery in a grid environment. Future Generation Computer Systems, 2011, vol. 27, no. 6, pp. 836-842
34. Yongcai Tao, Hai Jin, Song Wu, Xuanhua Shi. Scalable DHT- and ontology-based information service for large-scale grids. Future Generation Computer Systems, 2010, vol. 26, no. 5, pp. 729-739
35. Yiming Zhang, Xicheng Lu, Dongsheng Li. Survey of DHT topology construction techniques in virtual computing environments. Sci China Inf Sci, 2011, 54(11): 2221-2235

Figure 1. Example of the bloom filter calculation

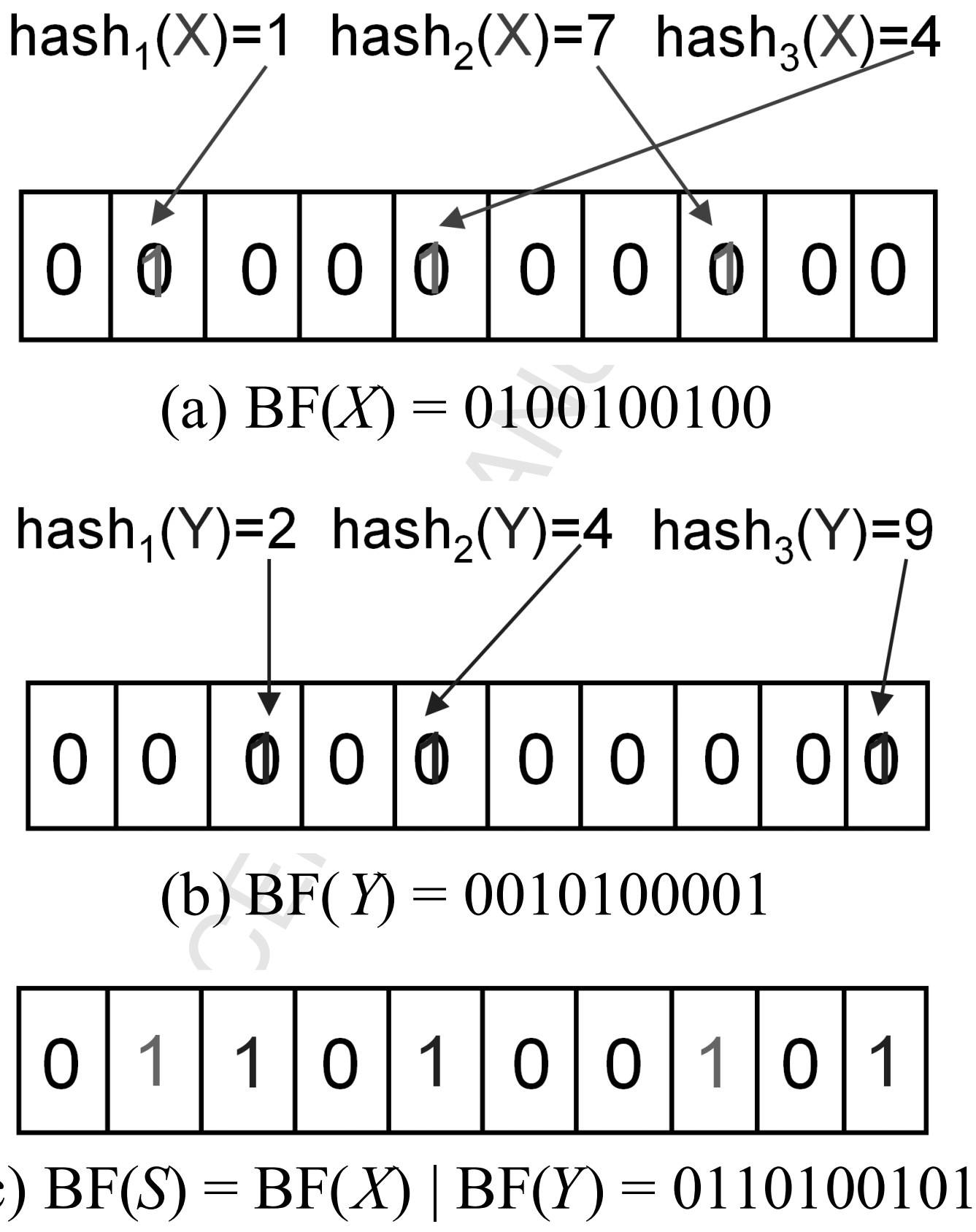


Figure 2. DABF-based hint dissemination and aggregation

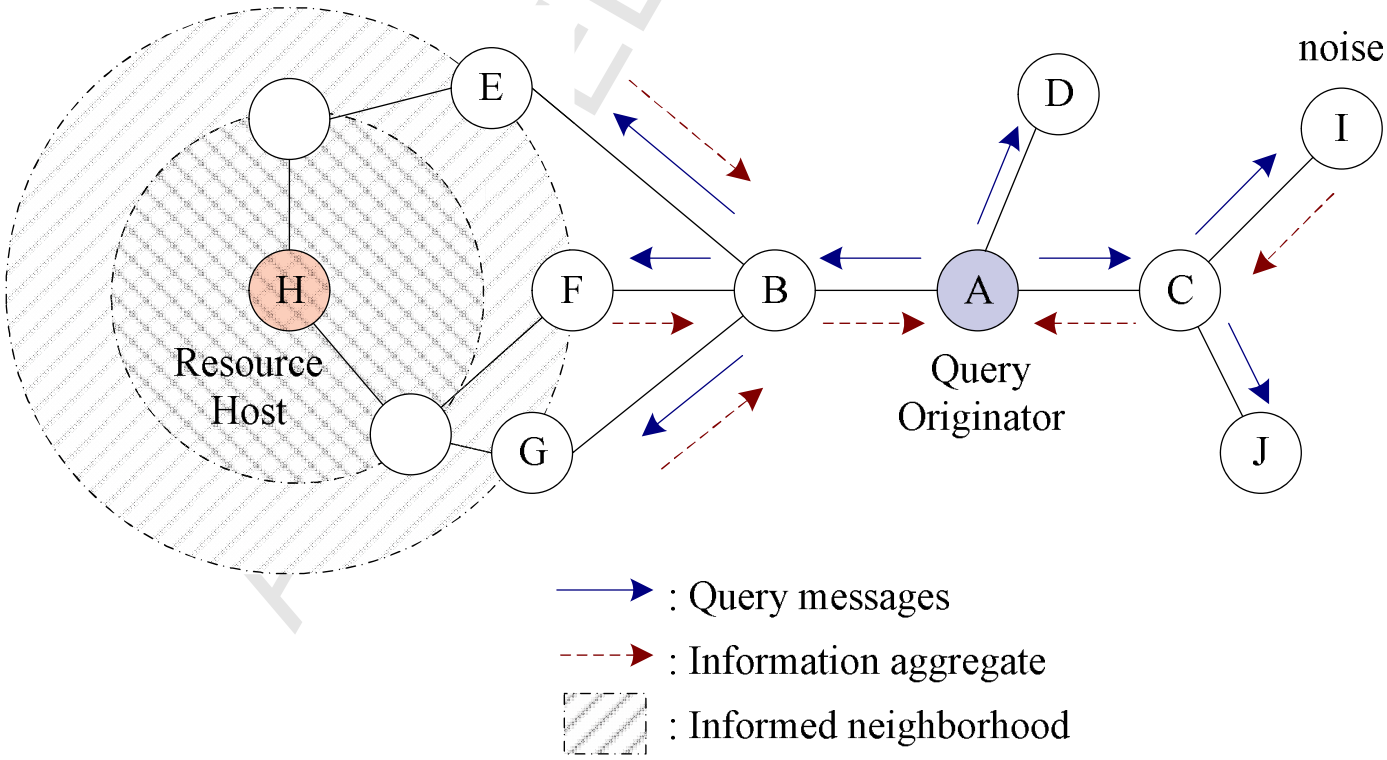


Figure 3. EDBF hint dissemination

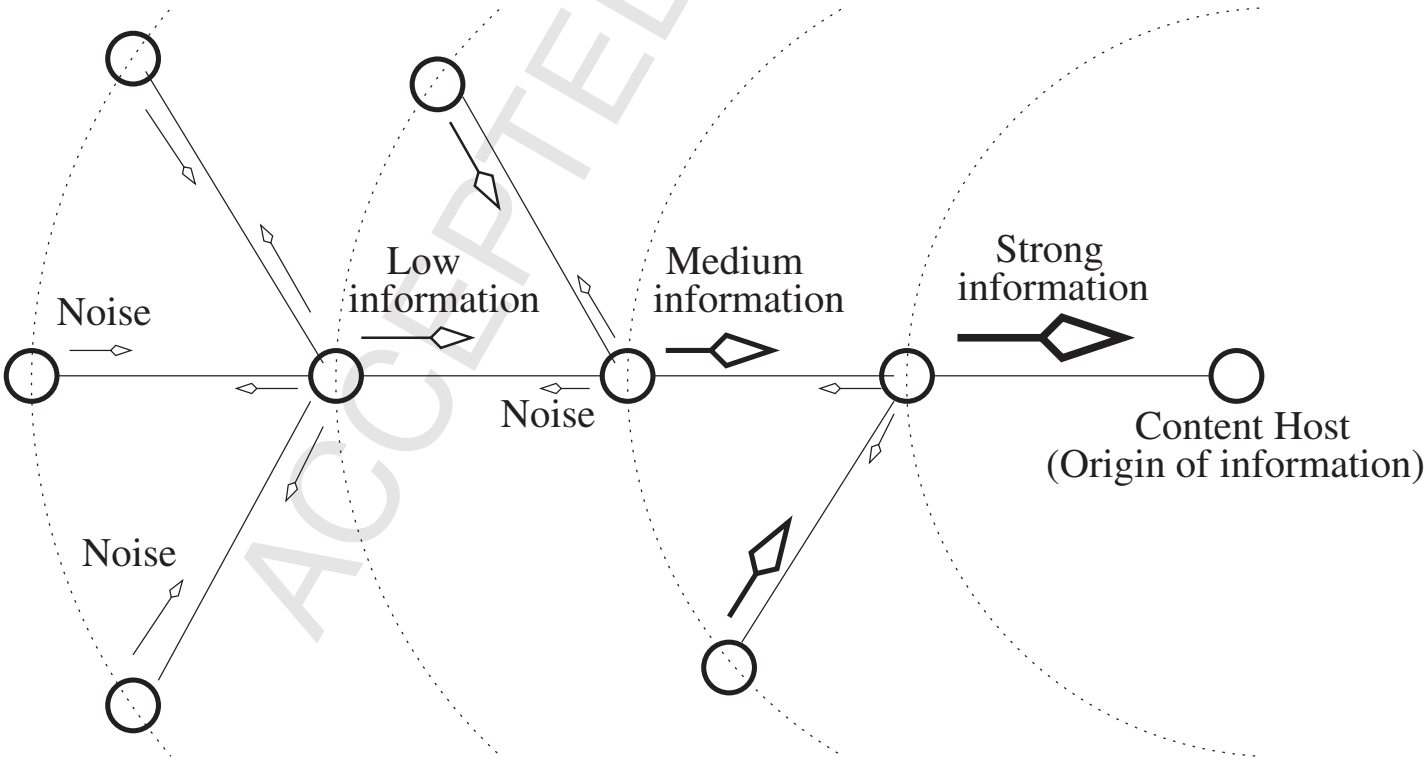


Figure 4. Search latency of CPS, SQR and Gnutella

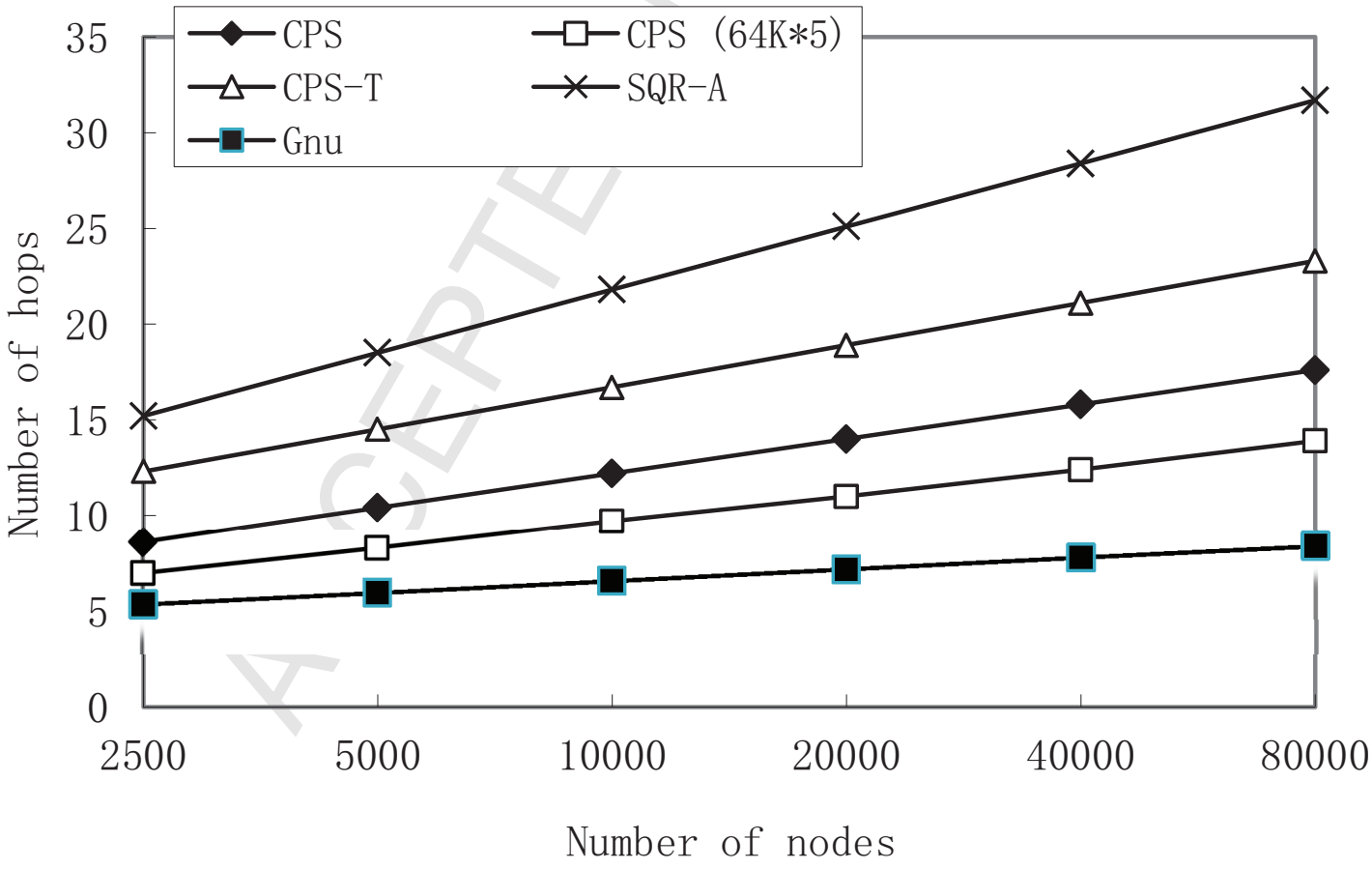


Figure 5. Message cost of CPS, SQR and Gnutella

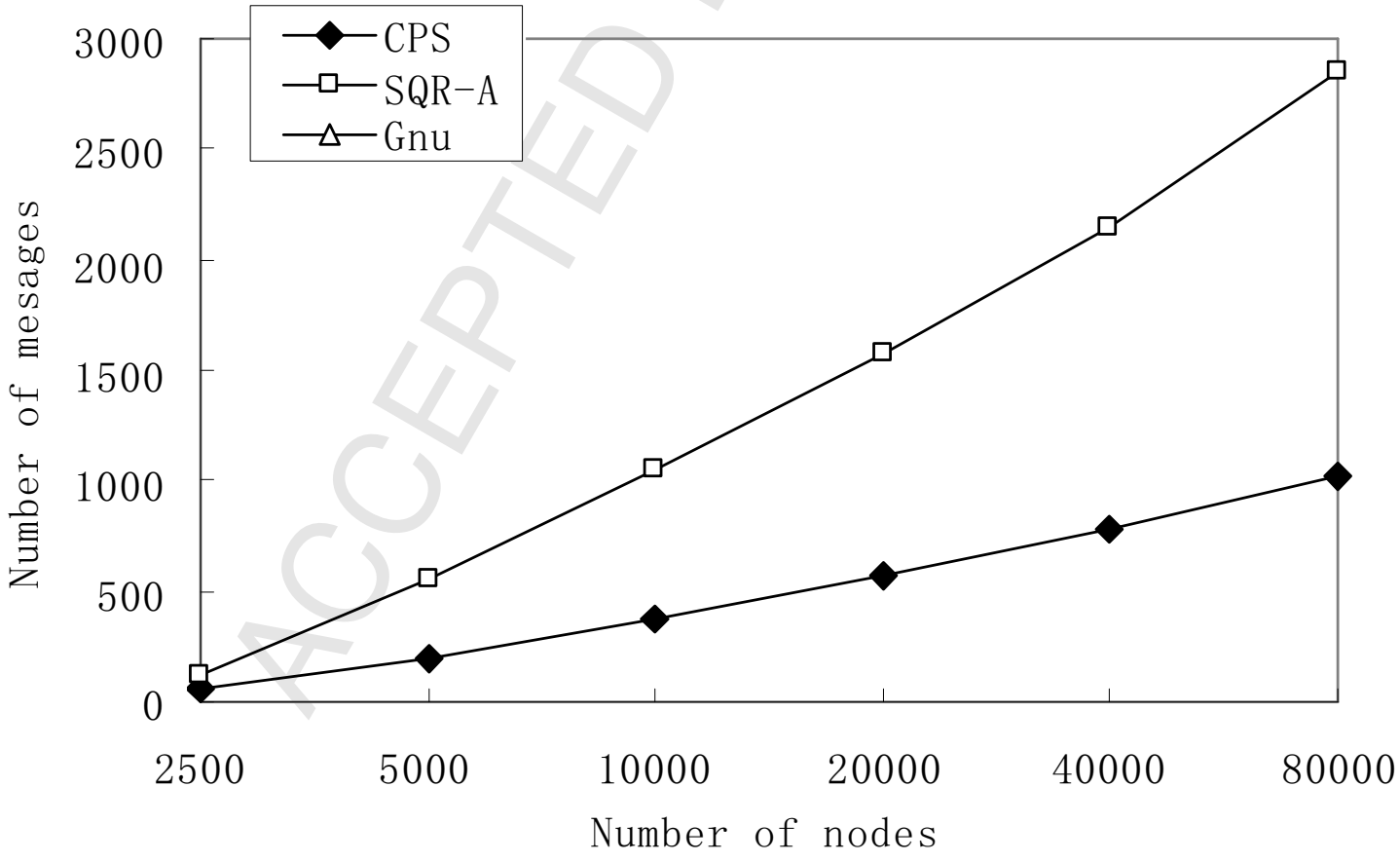


Figure 6. Impact of different parameter settings

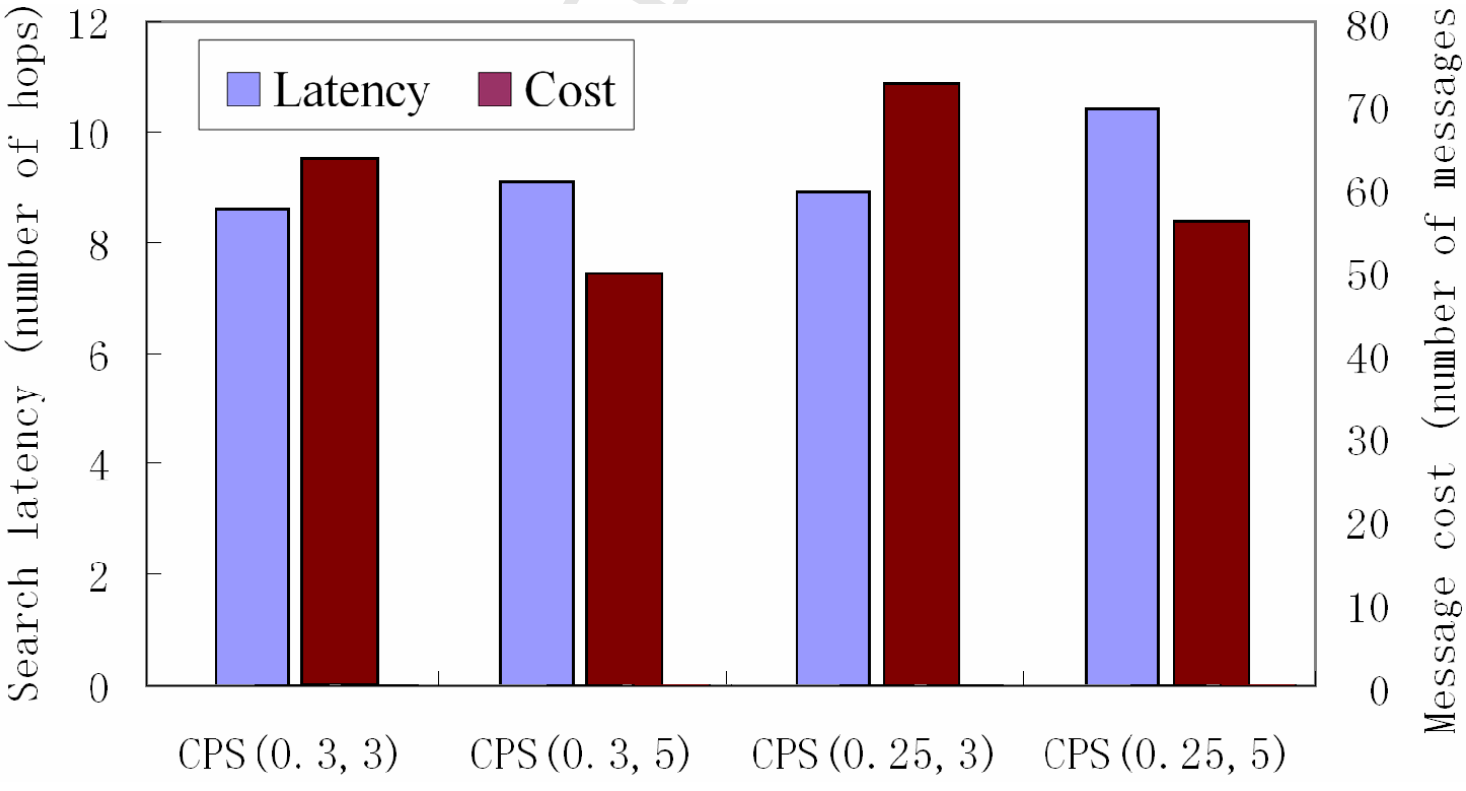


Figure 7. Robustness of CPS

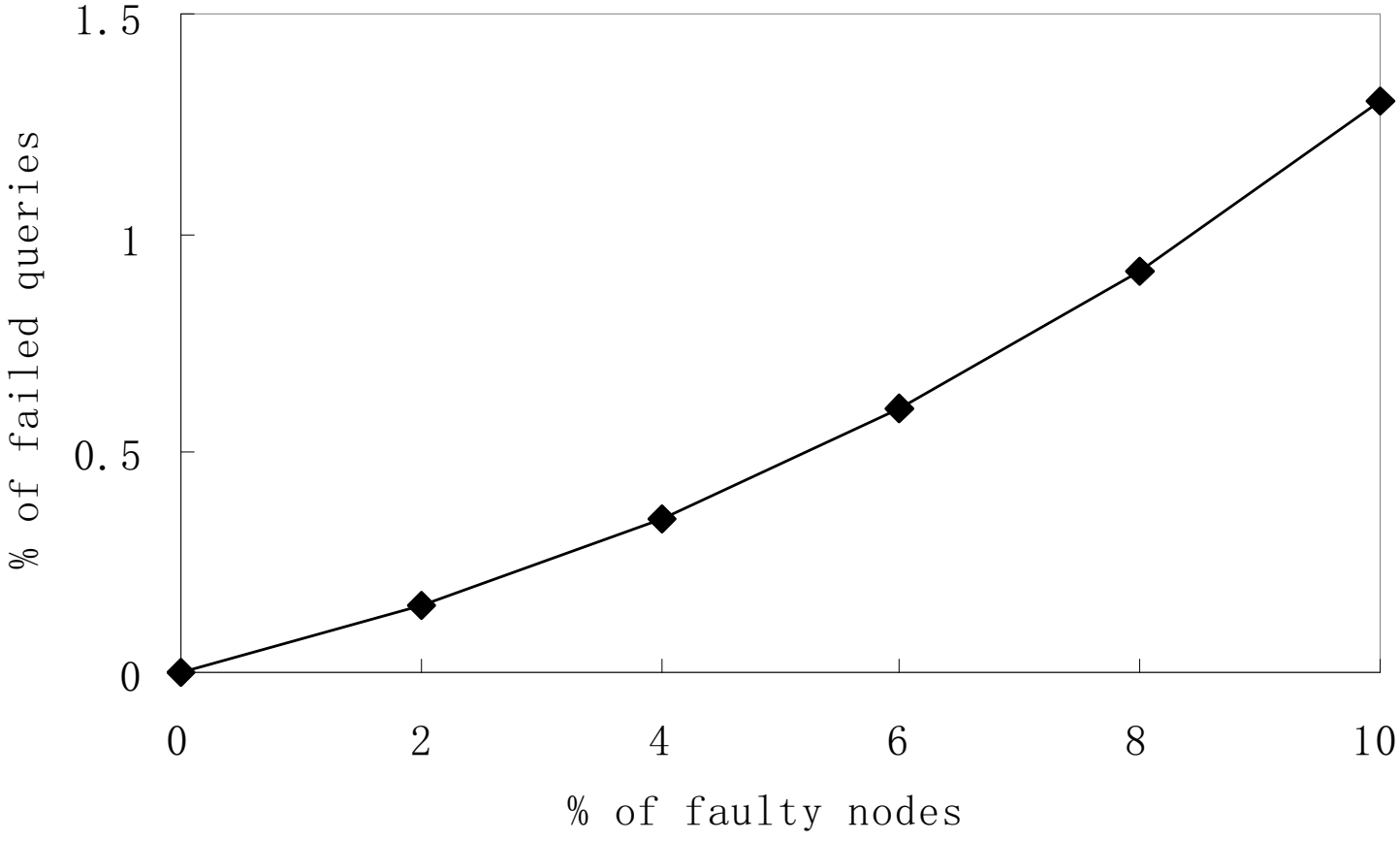


Figure 8. Impact of replication on the latency and cost of CPS

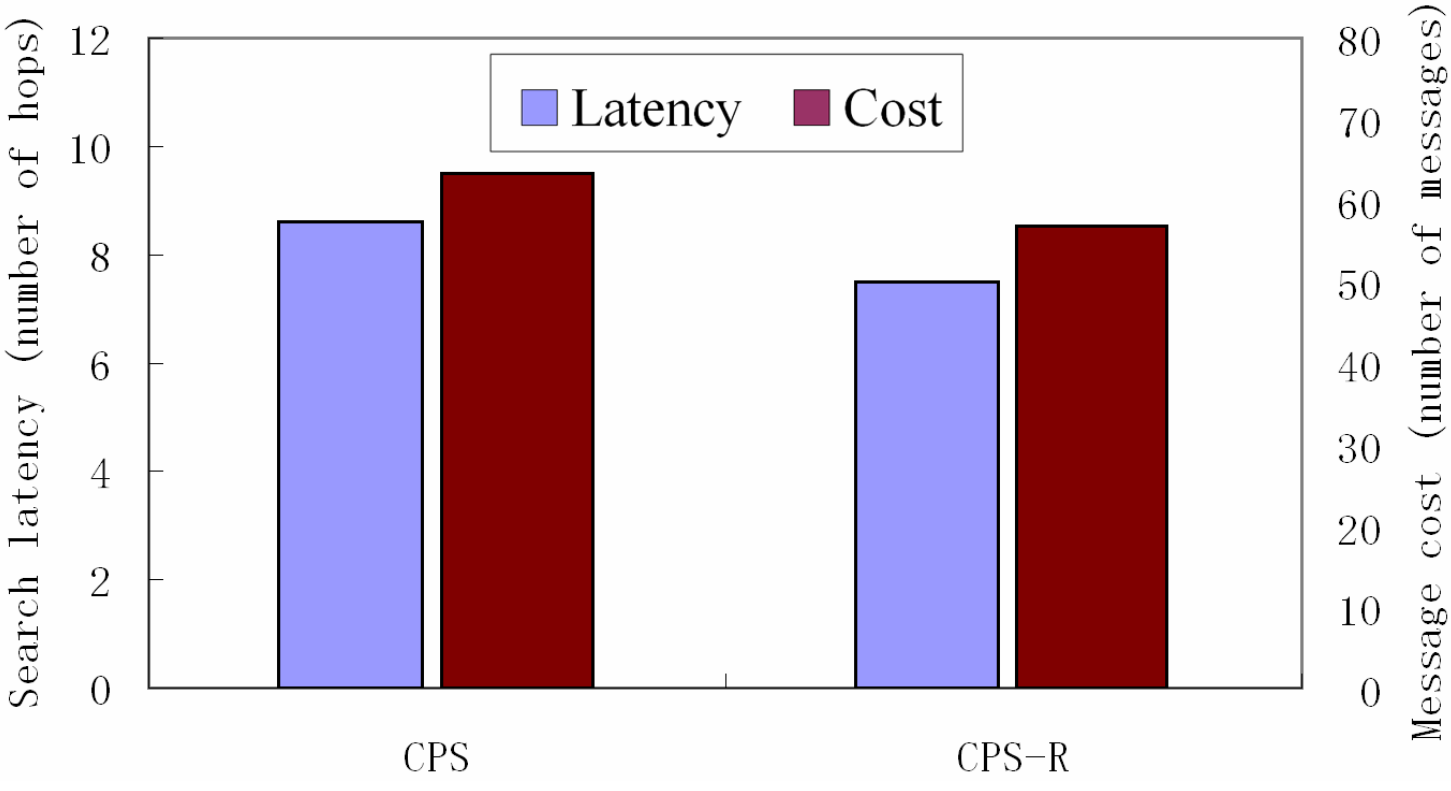


Table 1. PARAMETERS OF EXPERIMENTS

Parameter	Value
Common	
No. of Nodes	2,500 ~ 80,000
Average node degree	4
Available resources	10,000
No. of resources per node	2
No. of hash functions	64
CPS	
Width of Bloom Filter	64K bits, 64K × 5 bits
Mean value of log (*p*)	log 0.3
Fault-tolerant factor	0.5
No. of initial search units	3
Search acceleration period	3
No. of sub-SUs for acceleration	3
Thresholds	$Th_1 = 1/4$, $Th_2 = 1/16$
Column number in *BF* table	5
SQR-A	
Width of Bloom Filter	64K × 5 bits
Factor	0.3

Table 1. Parameters of Experiments

Yiming Zhang received the BSc degree and M.Sc. degree in Mechanics Engineering in 2001 and 2003, and the Ph.D. degree in Computer Science in 2008, from National University of Defense Technology (NUDT), Changsha, Hunan, China. He is now an assistant Professor in the National Laboratory for Parallel and Distributed Processing, NUDT. His research interests include distributed computing, peer-to-peer computing, and operating systems. He received the China Computer Federation (CCF) Best Dissertation Award in 2011 and the HP Distinguished Chinese Student Scholarship in 2008. Dr Zhang's current research is primarily sponsored by the National Natural Science Foundation of China (NSFC).

Ling Liu is a full Professor in the School of Computer Science at Georgia Institute of Technology. There she directs the research programs in Distributed Data Intensive Systems Lab (DiSL), examining various aspects of data intensive systems with the focus on performance, availability, security, privacy, and energy efficiency. Prof. Liu and her students have released a number of open source software tools, including GTMobiSim, PeerCrawl, WebCQ, XWRAPElite. Prof. Liu has published over 300 International journal and conference articles in the areas of databases, distributed systems, and Internet Computing. She is a recipient of the best paper award of ICDCS 2003, WWW 2004, the 2005 Pat Goldberg Memorial Best Paper Award, and 2008 Int. conf. on Software Engineering and Data Engineering. Prof. Liu has served as general chair and PC chairs of numerous IEEE and ACM conferences in data engineering, distributed computing, service computing and cloud computing fields and is a co-editor-in-chief of the 5 volume Encyclopedia of Database Systems (Springer). She is currently on the editorial board of several international journals, such as Distributed and Parallel Databases (DAPD, Springer), Journal of Parallel and Distributed Computing (JPDC), IEEE Transactions on Service Computing (TSC), ACM Transactions on Web (TWEB) and Wireless Network (WINET, Springer). Dr. Liu's current research is primarily sponsored by NSF, IBM, and Intel.

***Biographies (Yiming's Photograph)**
Click here to download high resolution image

***Biographies (Ling's Photograph)**
Click here to download high resolution image

Highlights

- Decaying BF with different deterministic masks for faraway information propagation.
- Intelligent message behaviors like suspend, resume, terminate, move, order, reside.
- Achieve a remarkable reduction in search latency and cost over previous approaches.