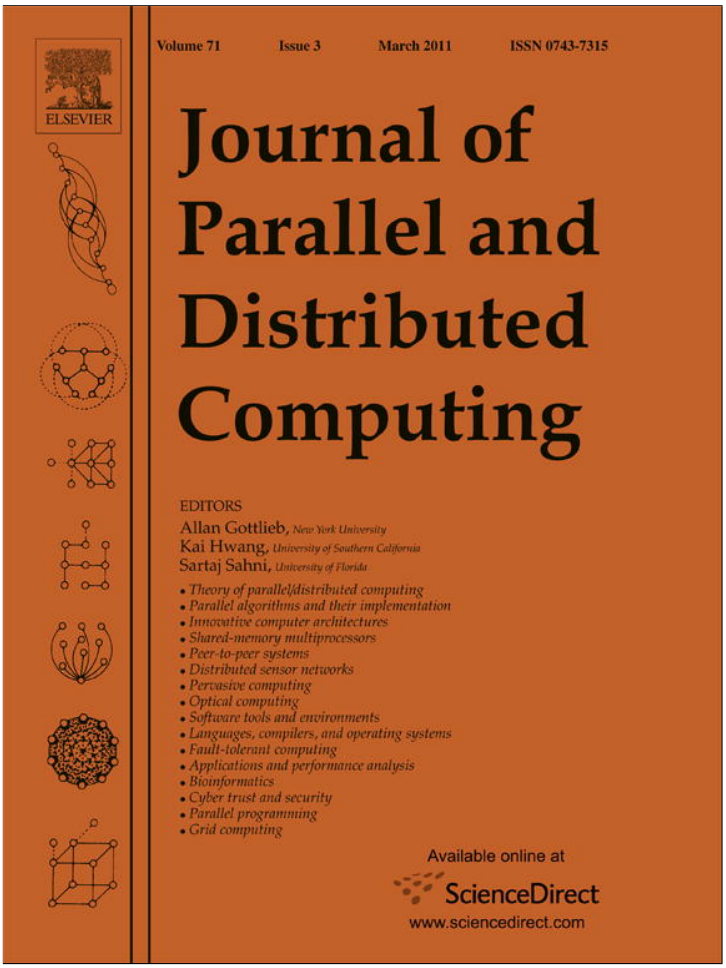


J. Parallel Distrib. Comput. 71 (2011) 407–423

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A flabellate overlay network for multi-attribute search

Ruixuan Li [a,*], Wei Song [a], Haiying Shen [b], Weijun Xiao [c], Zhengding Lu [a]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
[b] Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634, USA
[c] Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Received 22 January 2010
Received in revised form
24 September 2010
Accepted 5 November 2010
Available online 12 November 2010

Keywords:
Peer-to-peer network
Multi-attribute search
Range query
Virtual replica network
File replication

ABSTRACT

Peer-to-peer (P2P) technology provides a popular way of distributing resources, sharing, and locating in a large-scale distributed environment. However, most of the current existing P2P systems only support queries over a single resource attribute, such as file name. The current multiple resource attribute search methods often encounter high maintenance cost and lack of resilience to the highly dynamic environment of P2P networks. In this paper, we propose a Flabellate overlAy Network (FAN), a scalable and structured underlying P2P overlay supporting resource queries over multi-dimensional attributes. In FAN, the resources are mapped into a multi-dimensional Cartesian space based on the consistent hash values of the resource attributes. The mapping space is divided into non-overlapping and continuous subspaces based on the peer's distance. This paper presents strategies for managing the extended adjacent subspaces, which is crucial to network maintenance and resource search in FAN. The algorithms of a basic resource search and range query over FAN are also presented in this paper. To alleviate the load of the hot nodes, a virtual replica network (VRN) consisting of the nodes with the same replicates is proposed for replicating popular resources adaptively. The queries can be forwarded from the heavily loaded nodes to the lightly loaded ones through VRN. Theoretical analysis and experimental results show that FAN has a higher routing efficiency and lower network maintenance cost over the existing multi-attribute search methods. Also, VRN efficiently balances the network load and reduces the querying delay in FAN while invoking a relatively low overhead.

1. Introduction

In the past decade, peer-to-peer (P2P) technology has received a considerable amount of attention from the research and industrial communities. P2P enables one to search and fully utilize a variety of resources, such as computing, storing and networking resources, and file resources. However, most of the existing P2P systems only support content search over a single resource attribute (e.g. keyword or file name). Examples include the file sharing applications in Napster, Gnutella and BitTorrent. Full-blown P2P applications require an efficient resource content search over multi-dimensional resource attributes. For example, in a typical file sharing system, users may not only want to query by file name, but also by file type, file size, last-modified time, owner, feedback, and reputation degree. Thus, an efficient P2P resource search over multi-dimensional attributes is increasingly needed with the dramatic development of P2P applications.

Recently, several works have been proposed to achieve multi-dimensional attributes' resource search. CAN distributed hash table (DHT) [18] is a typical structured P2P overlay supporting resource search over multi-dimensional attributes. However, the maintenance cost of CAN is considerably high, especially when peers frequently join and leave the network due to zone mergence and division [18]. In order to reduce the maintenance cost, Shu et al. [21] and Schmidt and Parashara [19] use space-filling curves, such as z-curve and Hilbert Curve, to divide CAN zones. However, using space-filling curves may incur zone density for popular resources and affect the load balance of the DHT. Locher et al. [16] proposes eQuus using node cliques to reduce the maintenance cost. Many improved routing algorithms (e.g., range query [7], KNN query [15]) over CAN have been proposed for complex queries. However, these routing algorithms inherit the inherent weakness of CAN. Other researches, such as Mercury [3], PHT [5], DST [29], and m-Light [25], have been conducted to address the P2P resource search over multi-dimensional attributes. Nevertheless, these methods mainly use high overhead index structures over DHT overlay to support multi-dimensional search, rather than addressing the underlying DHT overlay problems. These approaches often encounter high maintenance cost and lack resilience to the highly dynamic environment of P2P networks,

* Corresponding author.
E-mail addresses: rxli@hust.edu.cn (R. Li), weisong@smail.hust.edu.cn (W. Song), shenh@clemson.edu (H. Shen), wxiao@umn.edu (W. Xiao), zdlu@hust.edu.cn (Z. Lu).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.11.002

especially when the network size is large, the dimension of resource attributes are very high, and peers join and leave the network rapidly.

The goal of our research is to develop a scalable P2P overlay supporting resource search over multi-dimensional attributes with high routing efficiency, low maintenance cost, and a fine load balance. Towards this end, we propose a Flabellate overlAy Network (FAN) system including FAN network construction, query routing and peer joining and leaving algorithms. Peers in FAN can find their desired resources in $O(\log(N/k))$ hops when the network size is N and each subspace can contain up to k peers. Joining and leaving the FAN network generates up to $O(\log(N/k))$ messages. Furthermore, we develop a range query algorithm over the FAN overlay. This uses resource distance and attribute slope to implement a range query that has a high routing efficiency and low maintenance cost.

In this paper, we use files as an example for resources. In general, the distribution of resources and user accesses are unbalanced in a P2P network due to the variance of resource popularity. Some nodes are busy due to hosting many hot resources, while other nodes only receive few requests. This distribution imbalance degrades the quality of service of a P2P network through delayed querying, message forwarding and so on. To address this issue, we propose a virtual replica network (VRN) to adaptively replicate the popular resources along hot search paths in FAN. VRN pushes the popular resources closer to the consumers and forwards the queries to the least loaded node, leading to reduced querying delay and a balanced load distribution.

Experimental results show that FAN is an efficient P2P overlay supporting multi-dimensional resource attributes with low maintenance cost and high routing efficiency. The single-dimensional FAN achieves logarithmic resource search efficiency comparable to Chord. Furthermore, the experimental results demonstrate that the VRN replication algorithm can greatly reduce the querying delay in FAN under a heavy querying load. CAN-based routing algorithms supporting complex queries [7,15] and P2P routing algorithms independent of the underlying P2P framework [4] can be implemented over FAN to obtain an enhanced performance.

The main contributions of this paper are as follows.

- A P2P flabellate overlay that supports resource search over multi-attributes within $O(\log(N/k))$ hops.
- A self-maintenance algorithm that ensures the scalability of FAN and brings $O(\log(N/k)/k)$ messages for each node join or departure.
- A range query algorithm over FAN using resource distance and attribute slope to achieve an efficient and scalable searching performance.
- A virtual replica network that balances the load between nodes and employs the power-law links to efficiently inform the load information.
- Extensive experiments to show the good performance of FAN.

The rest of this paper is organized as follows. Section 2 discusses the related research in P2P network. Section 3 presents the details of the FAN network. Section 4 describes the adaptive replication network for FAN. Section 5 discusses and evaluates the performance of FAN and outlines its benefits in comparison with CAN, Chord and Mercury. Finally, Section 6 concludes this paper with a discussion of possible improvements to be considered as future work.

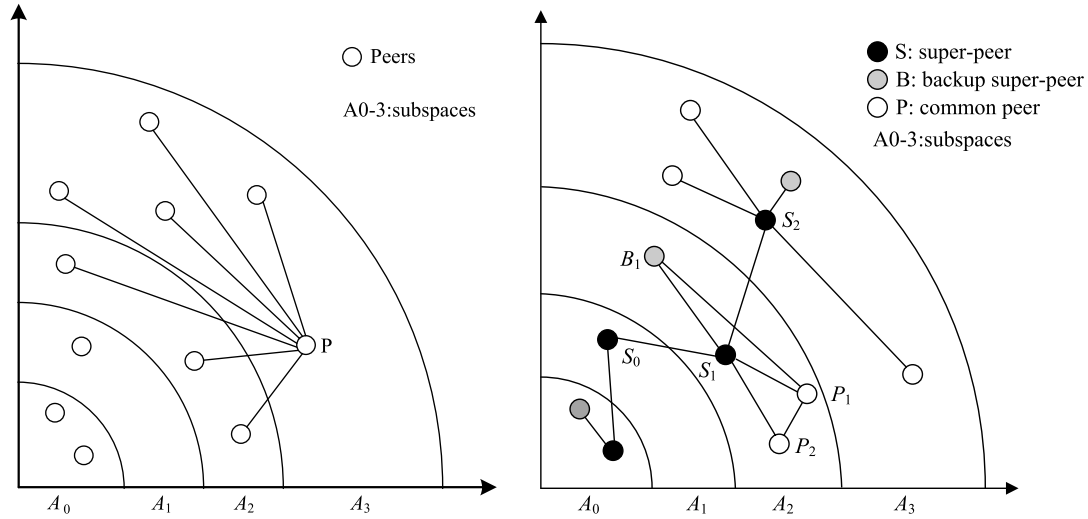
2. Related work

In the last few years, a number of methods have been proposed for P2P resource search. A routing algorithm is an essential part of a resource search. A structured P2P network employs DHT to build a scalable, load balanced network topology. A peer in structured P2P network has its own logical identifier and indexes the resources based on the identifier. The routing algorithms in structured P2P networks can be divided into two categories: (a) algorithms supporting single-dimensional resource attributes; (b) algorithms supporting multi-dimensional resource attributes.

Chord [23] and Tapestry [28] are structured P2P networks supporting resource search over a single attribute. The systems employ different topologies, such as ring and hypercube, and assign files to nodes using a consistent hashing function [13] to manage file resources. Their routing efficiency is $O(\log N)$. Many P2P applications need resources described by multi-dimensional attributes, thus there is a need for a search algorithm over multi-dimensional data. CAN [18] is a typical P2P overlay supporting multi-dimensional resource attributes. Each peer in CAN has a set of d-dimensional coordinates. Every node manages a virtual zone containing itself and stores their immediate neighbor's routing information. A node in CAN delivers the search messages towards the target peer through a simple greedy flooding approach. The CAN routing efficiency is $O(dN^{1/d})$, where d is the resource dimension. However, CAN lets a node hold several zones to deal with nodes leaving and the subsequent file resource takeover. Furthermore, when peers frequently join and leave or adjacent peers fail simultaneously, the maintenance cost of CAN is considerably high.

Recently, many works have been proposed to support P2P resource search over multi-dimensional attributes. Most of them are over DHT overlays. For example, Liu proposes an NR-tree [15] by adapting an R*-tree, Mercury [3] uses a multi-Chord structure, pSearch [24] employs vector space model (VSM) and latent semantic indexing (LSI) over CAN. VBI-Tree [12] and BATON [11] build tree indexing structures over P2P networks to support multi-dimensional range queries. Distributed segment tree (DST) [28], range search tree (RST) [9] and m-Light [25] employ tree indexes by placing data in internal nodes to perform multi-dimensional range queries with a balanced load distribution. P-Ring [6] uses a ring index to achieve scalable and robust range querying. Q-Tree [2] is a range querying tree index that can answer complex queries rapidly. Although these search methods are built over a multi-dimensional DHT overlay, they are designed to supply complex queries rather than improving the efficiency of multi-dimensional routing of the underlying P2P overlay. Hence, efficient search with multi-dimensional attributes is exactly the motivation for the design of FAN. Many improved routing algorithms over DHT overlay, such as [7,15], and other algorithms independent of underlying DHT topology, such as [4], can be implemented over FAN to achieve enhanced performance.

The variance of a resource's popularity and the skew of the user's access make some nodes become heavily loaded. There are many efforts [8,26,30] that address the issue of relieving the hot nodes by replicating the popular resources in P2P networks. Ganesan et al. [8] proposed to dynamically adjust data partitions based on data and access skew to achieve load balance. Shen and Xu [20] presented locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. Wang et al. [26] proposed to replicate hot resources in a content distribution network to improve search efficiency. Zhu and Hu [30] utilized the proximity information to guide load balance over structured P2P networks. The load balancing method in FAN distinguishes itself from the previous works by taking advantage of the routing in FAN. In FAN, the query routing messages are forwarded along the links of extended adjacent subspaces. FAN creates replicas of hot resources along routing paths based on the access skew to achieve an improved performance of load balance.



(a) FAN without super-peer. (b) FAN with super-peers.

Fig. 1. A 2-dimensional FAN network structure.

3. FAN protocols

3.1. The basic FAN network architecture

The shared resources in FAN [22] are described by d-dimensional attributes. The FAN uses the consistent hashing functions to compute the hashed values of resource attributes and maps the resource into a node of a d-dimensional Cartesian space. We call this d-dimensional Cartesian space the FAN mapping space. In the d-dimensional FAN mapping space, given a node P at the point $(x_1, x_2, \ldots, x_d)$, where $x_1, x_2, \ldots, x_d$ are the hash values of different resource attributes of P, we define P's distance D_p as its second moment to the origin of coordinates which is shown in Eq. (1). In FAN, both the resource search and network management are based on the peer's distance.

$$D_p = x_1^2 + x_2^2 + \cdots + x_d^2 = \sum_{i=1}^{d} x_i^2. \tag{1}$$

FAN mapping space is divided into non-overlapping and continuous subspaces based on the peer's distance. Each node falls into a unique subspace which covers its distance. The FAN subspace is defined as follows. We use $A(a, b)$ to denote a subspace which covers the peer's distance range $(a, b]$. For each node P in the subspace $A(a, b)$, its distance D_p satisfies $a < D_p \leq b$. Furthermore, each node P whose distance satisfies $a < D_p \leq b$ belongs to the subspace $A(a, b)$. To describe the FAN search algorithm easily, we define the distance between peer P and a subspace $A(a, b)$ in Eq. (2).

$$DS_{PtoA} = \begin{cases} a - D_P & a > D_P \\ 0 & a \leq D_P \leq b \\ D_P - b & D_P > b. \end{cases} \tag{2}$$

We draw a 2-dimensional FAN structure in Fig. 1(a). The resource search in FAN is equal to finding its subspace. Therefore, a peer in FAN must store the peer routing information in its same subspace and its adjacent subspaces. For example, in Fig. 1(a), peer P stores the information of all nodes in subspaces A_3 and A_2. This network structure makes peers in FAN store too much routing information. To reduce the routing information at each peer, we employ the super-peer technique to manage the subspaces. Fig. 1(b) illustrates a 2-dimensional FAN structure with super-peers. That is, the super-peers in neighboring subspaces are connected to each other, and each super-peer connects to all nodes in its own subspace.

In FAN, super-peers manage the subspaces and process search requests. Therefore, the super-peers play a more important role than common peers. To achieve good performance and high stability, a super-peer should have more computing power and a higher network bandwidth than the common peers. With more peers joining the subspaces, it is possible that some common peers may exhibit a better performance in CPU circles and network speed than the current super-peer. We assume that every peer in FAN is altruistic. The super-peer periodically checks all peers in the subspace to see if there is any common peer having more powerful computing and networking performance. If found, the powerful common peer will replace the super-peer to manage the subspace.

According to the related analysis [27], the redundant super-peers are needed for a realistic P2P system. Therefore, each subspace in FAN has a redundant backup super-peer watching the super-peer and synchronizing its routing information. When the current super-peer leaves, the backup one will replace it. The simultaneous leave of super-peer and backup super-peer will be discussed in Section 3.5. Every peer in FAN belongs to a subspace and maintains a routing table which stores the peer identifier, peer coordinates, and the subspace information described by the lower and upper distance borderlines. Table 1 shows the routing table information in the super-peer S_1 and the common peer P_1 in Fig. 1(b). The super-peer stores the information of all peers in its subspace, e.g. the first three rows in Table 1(a), and all super-peers in its adjacent subspaces, e.g. the last two rows in Table 1(a). At the same time, a common peer only stores the information of peers in its subspace, as shown in Table 1(b).

3.2. Resource search in FAN

In FAN, the shared resources and the peers are mapped into subspaces. Therefore, retrieving resources and searching peers in FAN is equivalent to finding the subspaces.

The super-peers in FAN take charge of managing the subspaces and processing the received search requests. While peer P searches resource R, P first computes R's coordinates and retrieves R locally. If R is found, the search is completed. Otherwise, P forwards the query to its super-peer S (if P itself is a super-peer, this step will be skipped). When S receives the search request, it asks the nodes in its subspace whether they have R. If the subspace contains R, the query will be satisfied. Otherwise, S forwards the query message to the super-peer of an adjacent subspace closer to R until the query

Table 1
FAN routing table.

Peer identifier	Peer coordinates	Subspace range
(a) Routing table in the super-peer $S1$		
B_1	(0.5, 3)	(5, 10)
P_1	(2.5, 1.2)	(5, 10)
P_2	(2.7, 1)	(5,10)
S_0	(1, 1.5)	(3, 5)
S_2	(2, 3)	(10, 18)
(b) Routing table in the common peer $P1$		
S_1	(2, 2)	(5, 10)
B_1	(0.5, 3)	(5, 10)
P_2	(2.7, 1)	(5, 10)

message reaches the subspace containing R. Thus, the FAN routing strategy always transfers the query message to a super-peer closer to the target. Algorithm 1 describes how to search resources in FAN.

Algorithm 1. Resource search in FAN.

Input: p, query source peer; r, target resource.
Routing (p, r)
1: p uses the hash function to map r into a node n;
2: **if** (n is in p's subspace) **then**
3: **if** (r is in P's local resource list) **then**
4: return r;
5: **else**
6: return null;
7: **end if**
8: **else**
9: **if** (p is a common peer) **then**
10: p delivers the query to its super-peer;
11: **else if** (p is a super-peer)
12: p delivers the query request to the nearest super-peer;
13: **end if**
14: **end if**
15: **end if**

We suppose that a subspace contains up to k peers. Then the number of subspaces in FAN is $O(N/k)$, which will be demonstrated by the simulations in Section 5. According to the resource search process described in Algorithm 1, the routing messages can only be relayed from one subspace to its immediate neighbors. Therefore, the routing efficiency of the basic search algorithm of FAN is $O(N/k)$, which is not so satisfactory. Like Chord, the key to optimizing FAN is the selection of long distance links in addition to the immediately adjacent subspaces. We introduce the extended adjacent subspace to improve the resource search efficiency in FAN.

3.3. The improved FAN network architecture

In the improved FAN overlay, a super-peer stores not only the immediate neighbor super-peers but also the super-peers in the subspaces at intervals of 2^j layers, which are defined as the extended adjacent subspaces. Fig. 2 illustrates an improved FAN overlay structure. In Fig. 2, subspaces A_0 and A_2 are two immediate neighbor subspaces of A_1, and A_3 is one extended adjacent subspace of A_0. In the improved architecture, a super-peer forwards the query message to the super-peer whose subspace has the shortest distance to the target instead of transferring it to its immediate neighbor subspace one by one. Fig. 3 gives a simple example of resource routing in an improved 2-dimensional FAN.

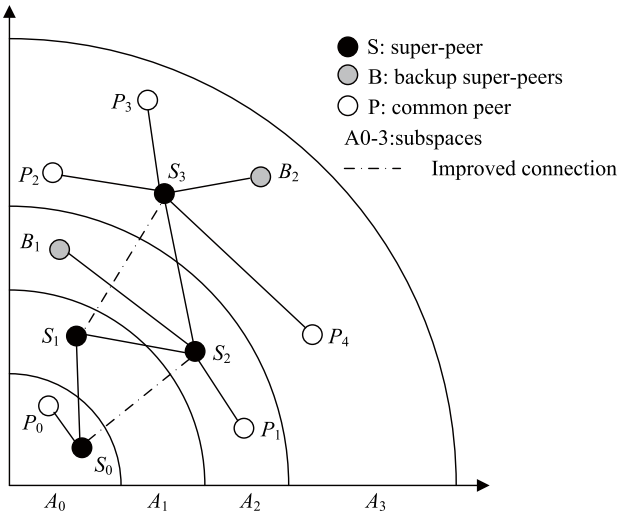


Fig. 2. The improved FAN structure.

Suppose that a query source peer and the target resource are at the intervals of M layer subspaces and every FAN subspace covers an equal distance range. By the links of the extended adjacent subspaces, the distance between the source peer and the target peer can be halved during a query message forwarding. Furthermore, M follows the uniform distribution $O(N/k)$. Therefore, a query message can reach its target in $O(\log(N/k))$ hops. By employing the extended adjacent subspaces, the efficiency of FAN's resource search has been improved from $O(N/k)$ to $O(\log(N/k))$.

However, this approach will introduce the additional cost of maintaining the extended adjacent subspaces. The extended adjacent subspace in FAN possesses an important property, the transitivity property. This will facilitate updating the information of the extended adjacent subspaces for super-peers.

Property 1 (*Transitivity*)**.** *If a subspace B_2 is an extended adjacent subspace of the subspace B_1 at the interval of 2^j ($j > 1$) layers, there must exist a subspace B between B_1 and B_2 with B being both B_1's and B_2's extended adjacent subspace. We call this property the transitivity of the extended adjacent subspace.*

Proof. The transitivity property is equivalent to the proposition that for any integer j bigger than 1, there must exist a mathematical decomposition of $2^j = 2^m + 2^n$. For any integer j bigger than 1, we can express 2^j as $2^j = 2^{k+1} = 2^k + 2^k$. □

Therefore, the transitivity property of extended adjacent subspaces is proved.

The peers in FAN, including super-peers and common peers, will periodically explore all the peers in their routing tables to update routing information. For the transitivity property of the extended adjacent subspaces, a super-peer only needs to know the immediate neighbor subspace information and periodically explores the known extended adjacent subspaces. This way, it can retrieve and update the information of all its extended subspaces.

By introducing the extended adjacent subspaces, a peer who receives a resource search request will first retrieve its local routing table. If any subspace A whose distance to the target node Q is 0, the target node is placed in the subspace A. Then the search request will be delivered to A's super-peer directly. Otherwise, the search request will be delivered to the nearest subspace to Q until it reaches the super-peer whose subspace covers Q.

In FAN, the super-peer stores the extended adjacent subspace information to reduce the load of the common peers. Since the super-peer stores more extended adjacent subspace information that enables it to quickly deliver the search requests to improve

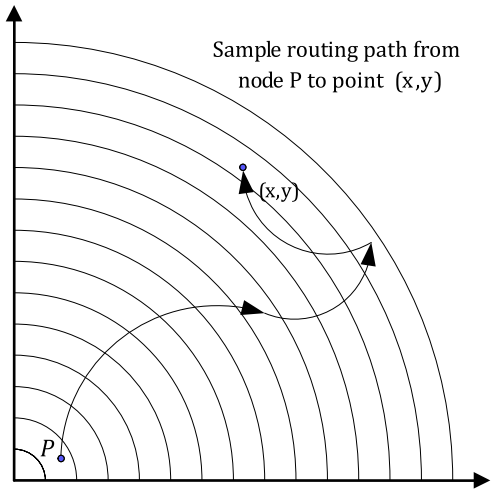


Fig. 3. An example of routing in a 2-dimensional FAN.

FAN search efficiency, the overhead should be controlled appropriately. In Section 5, an experiment has been carried out to verify whether the extended adjacent subspace information will overburden the super-peers.

3.4. Peer joining

When a peer P attempts to join in FAN, P firstly selects a node in the mapping space randomly and connects to a peer in FAN as the bootstrap. Afterwards, P routes to the subspace A which covers P, and attempts to join the subspace A. If the current number of peers in A does not reach k, P registers at A's super-peer to complete the joining. Otherwise, we adjust the subspace A by moving some peers of A to its immediate neighbor subspace rather than splitting it immediately since the FAN search efficiency is relative to the number of subspaces. In such situation, P first registers at A's super-peer S. Then S checks whether the number of peers in the two immediate neighbor subspaces A_1 and A_2 has reached k. If both subspaces A_1 and A_2 are full, we split the subspace A into two new subspaces with an equal number of peers. Otherwise, in order to decrease the number of FAN subspaces, we adjust the border of subspace A to make its immediate neighbor subspace, which has fewer peers, cover part of the peers originally in subspace A. The peer joining process in FAN is given in Algorithm 2. In Section 5, we have carried out an experiment to evaluate the effectiveness of the subspace management approach.

In FAN, we use the subspace adjustment method to decrease the number of subspaces. However, the super-peer may be transferred during the subspace adjustment operation. As shown in Fig. 4, we assume that a subspace can contain up to 4 peers. When a new peer joins subspace A, A adjusts the borderline with its immediately adjacent subspace B. After the adjustment, the former super-peer of A joins the new subspace B'. We call the transferred super-peer an outdated super-peer.

When the super-peer transfers in a peer joining process, the resource search requests will still be delivered to it before the new subspace information reaches all the extended adjacent subspaces. Therefore, the outdated super-peer does not discard the routing information immediately to achieve the stability of FAN resource search. The new selected super-peer (S_1') gets the routing table information from the outdated super-peer (O_1). Then S_1' sends an update message to all peers listed in its routing table to inform them of the change of super-peer in subspace A. While S_1' gets acknowledgements from all informed peers, it will send a message to the outdated super-peer O_1. Then O_1 could discard the outdated routing information in its routing table. When O_1 gets a search request before it discards the outdated routing information, it can simply deliver the request to the new super-peer S_1' and S_1' will process the search instead.

Algorithm 2. Peer joining in FAN.

Input: p, the peer that attempts to join FAN.
PeerJoin (p)
1: p connects to a peer as bootstrap;
2: p finds subspace A covering it by routing algorithm in Algorithm 1;
2: **if** (peer amount in $A < k$) **then**
3: p registers at the super-peer and creates the routing table to complete joining process;
4: **else if** (peer amount in both immediately adjacent subspaces reaches k) **then**
5: Split A into two subspaces and reconstruct the routing links for these two subspaces using Algorithm 4;
6: p joins one split subspace to complete joining process;
7: **else**
8: Adjust the border of A with its immediately adjacent subspace;
9: p joins one adjusted subspace to complete the joining process;
10: **end if**
11: **end if**

The messages generated during the joining process are computed in four cases as follows.

(a) When the number of peers in a subspace is less than k, the new joining peer needs to register at all peers in the subspace and construct its routing table to complete the joining. In this case, the joining process generates $O(k)$ messages.

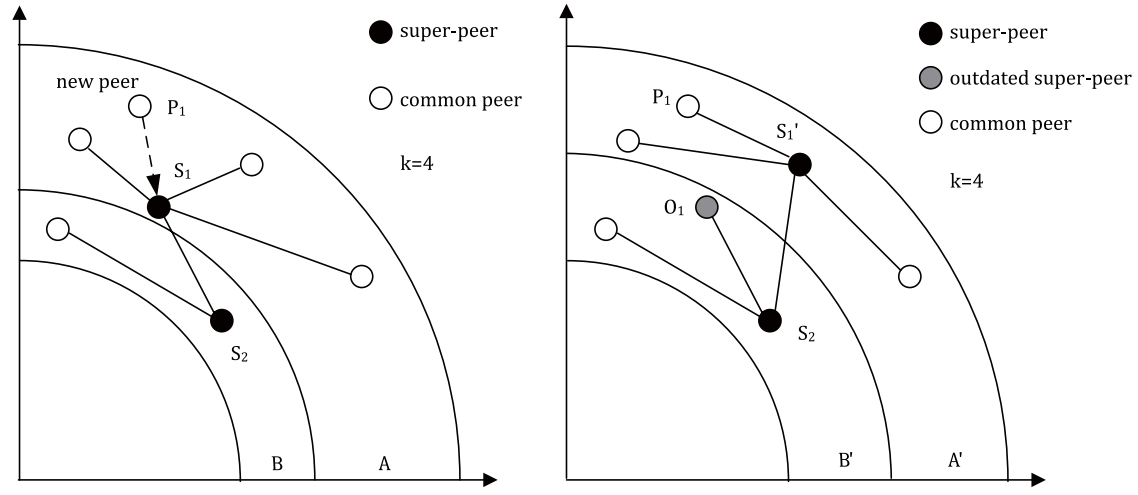
(b) When a joining process requires the subspace adjustment but no super-peer transferring, some common peers will change subspaces. We assume that the peers in a subspace follow the uniform distribution from 1 to k. Therefore, the mathematical expectation of the peer with a changing subspace is expressed in Eq. (3). Meanwhile, two adjusted subspaces should transmit the new subspace information to their $O(\log(N/k))$ extended adjacent subspaces. In this case, it generates $O(k/4+2\log(N/k))$ messages in total.

$$E(\text{peer}) = \left(\sum_{i=1}^{k-1}\left(k+1-\frac{k+1+i}{2}\right)\right)\Big/(k-1) = \frac{1}{4}\left(\frac{k^2+k-2}{k-1}\right) = \frac{k}{4}+\frac{1}{2}. \quad (3)$$

(c) If a super-peer transfers during the subspace adjustment, $O(k/4)$ peers will change their subspaces. The new super-peer needs to get the routing information from the former super-peer. The two super-peers need to issue the new subspace information to $O(\log(N/k))$ extended adjacent subspaces. In this case, it generates $O(k/4+2\log(N/k))$ messages in total as well.

(d) If both of the two immediate neighbor subspaces are full, the full subspace will be split into two new subspaces with equal number of peers. The new subspaces need to inform $2\log(N/k)$ extended subspaces with the new routing information. In this case, the joining process generates $O(2\log(N/k))$ messages.

By analyzing the above cases (a)–(d), we can find that the cost of a peer's joining process in FAN is $O(\log(N/k)/k)$. The expected number of messages in a joining process is expressed in Eq. (4).



(a) Before subspace adjustment. (b) After subspace adjustment.

Fig. 4. Super-peer changes when adjusting subspaces.

$$E(\text{cost}) = \frac{(k-1)\times k + \left(1-\frac{1}{k^2}\right)\left(\frac{k}{4}+2\log\left(\frac{N}{k}\right)\right)+\frac{1}{k^2}2\log\left(\frac{N}{k}\right)}{k} = k+\frac{2}{k}\log\left(\frac{N}{k}\right)-\frac{1}{4k^2}-\frac{3}{4}\approx k+\frac{2}{k}\log\left(\frac{N}{k}\right)-\frac{3}{4}. \quad (4)$$

3.5. Peer departing

When a peer leaves FAN, the associated routing information should be updated on time to ensure the integrity of FAN's structure. When a common peer attempts to leave FAN, it only needs to notify the peers in its subspace to complete the leaving process. In this case, it generates $O(k)$ messages. When a super-peer wants to leave, the backup super-peer will replace it if the leaving super-peer is not the last peer in the subspace. The message generated in this case is also $O(k)$. However, if the leaving super-peer is the last peer in the subspace, the immediately adjacent subspace with fewer peers will take over this empty subspace and $O(\log(N/k))$ messages will be generated. Therefore, it is found that the peer leaving process in FAN produces $O(\log(N/k)/k)$ messages which are expressed in Eq. (5).

$$E(\text{cost}) = \frac{(k-1)\times 1+2\log(N/k)}{k} \approx 1+\frac{2}{k}\log\left(\frac{N}{k}\right). \quad (5)$$

Algorithm 3 describes the peer leaving process in FAN. For the stability of FAN, any node's departure should not break the link of the extended adjacent subspace. However, the simultaneous departure of the super-peer and backup super-peer in a subspace will distort the extended subspace links. To maintain the stability of the FAN network and repair the extended subspace links in time after a simultaneous exit of both the super-peer and the backup super-peer, each common peer stores the information of all other peers in the same subspace and the super-peers in the immediately adjacent subspaces, as shown in Table 2. Comparing Table 2 with Table 1(b), P_1 stores additional routing information, including the super-peers in adjacent subspaces, such as the last two rows in Table 2. When a common peer detects that the super-peer and the backup super-peer are both offline in their periodical exploration, the common peer will notify other peers in the same subspace to choose one as the new super-peer, which has more computing power and network bandwidth. The new super-peer contacts the immediately adjacent subspaces to get its adjacent subspace information. This is because the neighbor subspace of a subspace S's extended adjacent subspace is both the extended adjacent subspace of an S's neighbor subspace. FAN subspaces can reconstruct the extended subspace links easily while the super-peer and the backup super-peer depart synchronously. Algorithm 4 describes the process of how to recover the extended subspace link in this situation.

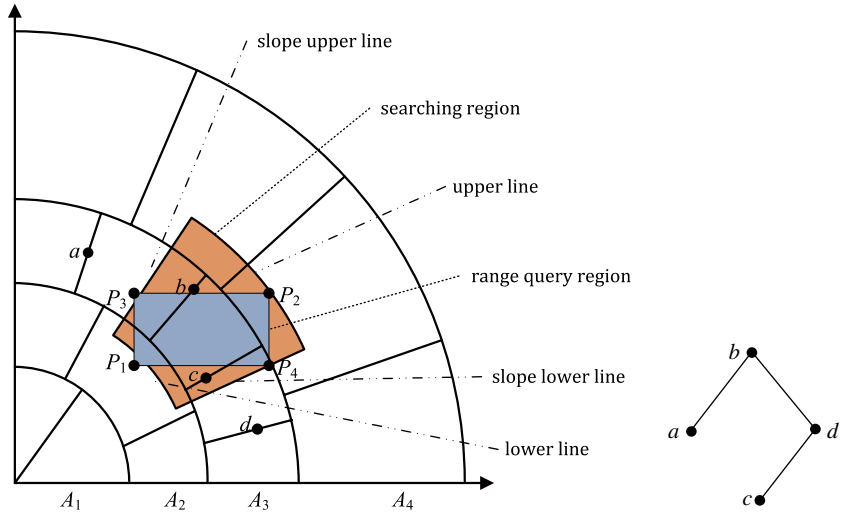
Table 2
The improved routing table in a common peer, including information on peers in its subspace and super-peers in its adjacent subspaces.

Peer identifier	Peer coordinates	Subspace range
S_1	(2, 2)	(5, 10)
B_1	(0.5, 3)	(5, 10)
P_2	(2.7, 1)	(5, 10)
S_0	(1, 1.5)	(3, 5)
S_2	(2, 3)	(10, 18)

3.6. Range query in FAN

In a multi-dimensional space, a range query denotes a multi-dimensional region and returns all resources falling in this region. In this section, we present a range query algorithm over FAN.

FAN overlay is built based on peer distance. To achieve the efficiency and flexibility for a multi-dimensional range query, we employ the resource distance and resource attribute slope to address this issue. As shown in Fig. 5(a), a 2-dimensional range query denotes a rectangular region covering several FAN subspaces, such as subspaces A_2, A_3, and A_4 in Fig. 5(a). For a range query, it has a lower peer distance node (P_1) and an upper peer distance node (P_2). We call the distance of these two peers as the lower distance line and upper distance line, respectively. Furthermore, for a multi-attribute FAN overlay, we select two attributes randomly to use their slopes in organizing the resources in a subspace. Then we use the attribute slope to describe a range query. Similarly, it has a lower slope line and an upper slope line, just as P_3 and P_4 in Fig. 5(a). Thus, all resources satisfying a range query are located among the lower distance line, the upper distance line, lower slope line and upper slope line. We call this region searching region for a range query. Other dimensional range queries are similar to the 2-dimensional example. However, not all resources in this searching region satisfy the range query. In the following example, we will introduce a searching tree index for each FAN subspace to filter the resources in the searching region.



(a) Range query in 2-dimensional FAN.

(b) Searching tree index of A_3.

Fig. 5. FAN range query.

Algorithm 3. Peer leaving in FAN.

Input: p, the peer that attempts to leave FAN
PeerLeave (p)
1: **if** (p is a common peer) **then**
2: p notices peers in the subspace or other peers detect p has left;
3: peers delete the leaving peer information;
4: return;
5: **else if** (p is a super-peer) **then**
6: **if** (backup super-peer b is online) **then**
7: p notices b or b detects super-peer has left;
8: b notices peers in the subspace and other super-peers in the extended adjacent subspaces to replace the super-peer and choose a new backup one;
9: **else if** (backup super-peer b is simultaneously offline) **then**
10: **if** (p is the last one in the subspace) **then**
11: an immediately adjacent subspace takes over the empty subspace and update the routing information;
12: **else**
13: reconstruct the routing link using Algorithm 4;
14: **end if**
15: **end if**
16: **end if**
17: **end if**
18: return;
19: **end if**

Algorithm 4. Recovering the extended subspace link when the super-peer and the backup super-peer leave the FAN simultaneously.

Input: p, the common peer detecting the offline of super-peer and backup super-peer
RecoverLink (p)
1: p notices other peers in its subspace;
2: the notified common peers elect a powerful peer p' as the new super-peer;
3: p' contacts the super-peers in its immediately adjacent subspaces to get their extended adjacent subspaces;
4: p' uses the immediately adjacent subspaces of these extended adjacent subspaces to reconstruct the routing link;

Algorithm 5. Range query in FAN.

Input: R, a range query;
RangeQuery(R)
1: route the query to an unsearched subspace A covered by R using Algorithm 1;
2: retrieve resources satisfying R in A using Algorithm 6;
3: **while** (the searching region of R is not completely covered by the searched subspaces) **do**
4: transmit R to A's neighbor subspace B whose distance range is between the lower line and the upper line of R;
5: retrieve resources satisfying R in B using Algorithm 6;
6: **end while**
7: return resources;

Algorithm 6. Subspace range query using searching tree index.

Input: R, a range query; *node*, root node of subspace searching tree index;
subSpaceRangeQuery(R, *node*)
1: **if** (*node.key* $\geq$ R.lower slope) **then**
2: subSpaceRangeQuery(R, *node.leftchild*);
3: **end if**
4: **if** (*node.key* $\leq$ R.upper slope) **then**
5: subSpaceRangeQuery(R, *node.rightchild*);
6: **end if**
7: **if** (*node.key* $\geq$ R.lower slope and *node.key* $\leq$ R.upper slope) **then**
8: return *node*'s resources;
9: **end if**

In a 2-dimensional FAN overlay, the searching region is described by the two attributes, as shown in Fig. 5(a). Each subspace has a searching tree index based on resource attribute slope, as shown in Fig. 5(b). The searching tree index is a *balanced search tree* in which a node denotes a resource and the key is the attribute slope. For each node in the subspace searching tree index, its left sub-tree consists of resources whose key is smaller than the node and its right sub-tree contains resources whose key is larger than the node. The searching tree index is maintained at super-peer in each subspace. There are many methods to maintain a balanced

search tree. Therefore, in this paper, we do not discuss how to maintain the searching tree index.

For a range query R, the initial node first routes it to a subspace A covered by R. Then the super-peer of A retrieves resources in the subspace using searching tree index based on R's slope range. Furthermore, if A's neighbor subspaces are also covered by R, A will forward the searching request to them to continue searching until the searching region of R is completely covered by the searched subspaces. The FAN range query algorithm is shown in Algorithm 5. For a range query R covering s subspaces, we suppose that each subspace contains t resources. Since the searching tree index is a balanced search tree, the searching cost of a range query in a subspace is $O(\log(t))$ and the whole range query cost is $O(s \bullet \log(t))$.

4. Adaptive replication algorithm for FAN

4.1. Popular resource replication and load balance

In a structured P2P network, each peer indexes some shared resources by the DHT function. The motivation of this resource management method is to balance the load among the peers and to improve the efficiency of the resource search. However, the distribution of resources and user accesses in a P2P network are usually unbalanced. Some peers who host the popular resources become hot nodes in the network. These hot nodes process most of the search requests in the network, while many other nodes process only a few search requests. As a consequence, the queues of search requests at the hot nodes are often full. Many query messages have to wait in the query queue for a long time at the hot nodes. In this situation, the search hops are less, yet the query delay is increased greatly. Hence, the hot nodes become bottlenecks for processing the search requests. To address this issue, we propose an adaptive replication algorithm over FAN for the popular resources to relieve the load of the hot nodes.

To decrease the query delay, FAN stores the popular resources closer to the requester by creating replicas of the popular resources along the search paths of hot resources. To replicate the most popular resources on the hot accessing path, a peer needs to analyze the popularity of its local resources, as well as the hot accessing path. In FAN, all query messages spread along the links of the extended adjacent subspaces. Therefore, a peer can easily discover the popularity of the shared resources and the hot search paths by analyzing the search records. Based on the structure of FAN, the super-peer of each subspace takes charge of analyzing the history of the search requests. A super-peer maintains a query message table to record the number of queries for each resource and the average query hops for each resource, as well as the last node the queries travel through before they arrive at the latest time interval.

A sample query message table is shown in Table 3. The query message table is sorted by the count of queries, which demonstrates the popularity of resources. Furthermore, each item is sorted by the number of queries from the last hop nodes. From the query message table shown in Table 3, we can conclude that the resource r_3 is the most popular, since in the last time interval the node has received the most (i.e. 120) queries for this resource. Furthermore, we can find that the hot accessing path for each shared resource; for example, the nodes n_1, n_2, and n_0 are the hottest accessing paths for the resources r_1, r_2, and r_3, respectively. If a resource needs to be replicated, we should create the replicas on the hottest accessing path.

By analyzing the recent access records in the query message table, a super-peer can easily discover the popular resources indexed locally and the hot accessing path. Then the super-peer selects the popular resource to replicate on the hot accessing path.

Table 3
A sample query message table at a node.

Resources (number of queries)	Last node traveling through				Average hops
$r_3(120)$	$n_0(50)$	$n_1(30)$	$n_2(20)$	$n_3(20)$	3.3
$r_2(90)$	$n_2(35)$	$n_1(30)$	$n_0(20)$	$n_3(5)$	2.8
$r_1(50)$	$n_1(20)$	$n_0(15)$	$n_2(10)$	$n_3(5)$	4.7

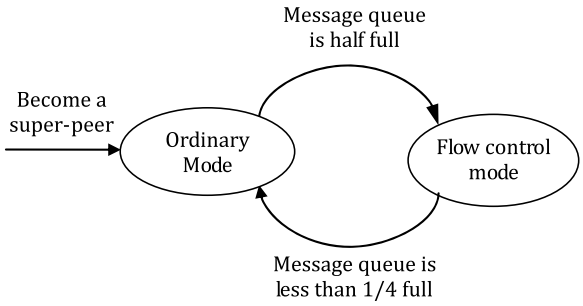


Fig. 6. Super-peer state transition graph.

We propose an adaptive replication approach to address this issue in this section.

To reduce the query delay in FAN, we employ a flow control method for the super-peer to restrict the length of the query message queue. Each super-peer has an FIFO (i.e. first in first out) queue to store the received query messages. When a node's query queue is half full, the node enters flow control mode. When the length of the query message queue is less than one quarter full, the node will turn back to its ordinary mode. The state transition for a super-peer is shown in Fig. 6.

When a super-peer S in flow control mode receives a query q, it calls the popular resource replication algorithm described in Algorithm 7 to process it. If q's forwarding count has reached a threshold *max_fwd*, we conclude that q has passed a long travel. Therefore, we give q a high priority, and evaluate whether q's target resources should be replicated. If the target resource r of the long traveling query q is a top-k popular local resource or r's average hops are bigger than a threshold *max_hops*, S creates the new replica via the function *new_replica* in Algorithm 8. Otherwise, S chooses the least loaded node n from r's virtual replica network (VRN) which is made up of the nodes that hold the replicas of the resource r. We will define the virtual replica network in Section 4.2. If n is also in flow control mode, n rejects receiving the new query. Then S inserts q into the local query queue and calls the function *new_replica* to replicate the resource r. Furthermore, a node periodically evaluates the recent received queries. If it finds that any resource replicas are little used, it drops them and leaves their VRNs.

Algorithm 8 describes the method for creating a new replica of a popular resource. Since the FAN queries are routed along the links of the extended adjacent subspaces, the replication algorithm chooses the hottest path the queries have traveled through to create the replica. Therefore, many queries can be satisfied on the halfway to the original target. Furthermore, we create the VRN for the popular resources in which busy nodes will forward the query to the node with the least load. As a consequence, a requester has a high probability of getting the desired resources in a shorter delay than before. We have developed the simulation experiments to evaluate the FAN query delay in Section 5.

4.2. Virtual replica network

The VRN of a resource r is a virtual network made up of the nodes which host the replica of r. As described above, load balance

Algorithm 7. Adaptive replication.

Input: q, query message; r, q's target resource
Replication(q)
1: **if** (q's forwarding count has reached *max_fwd*) **then**
2: **if** (r is top-k popular resource locally or r's average hops are bigger than *max_hops*) **then**
3: create new replica via function new_replica();
4: **return**;
5: **end if**
6: **end if**
7: finding the least loaded node n from VRN;
8: **if** (n is in flow control model) **then**
9: insert q into the query queue;
10: create new replica via function new_replica();
11: **else**
12: q's forwarding count++;
13: forward q to n;
14: **return**;
15: **end if**

Algorithm 8. Creating new replica.

new_replica()
1. find the node $n \notin$ VRN from query message table with the maximum query message;
2. **if** found **then**
3. create a new replica at node n;
4. n joins the VRN;
5. **else**
6. randomly select a node n' at the average hops from FAN;
7. create a new replica at node n';
8. n' joins the VRN;
9. **end if**

in FAN is achieved by forwarding the queries from the heavily loaded nodes to the lightly loaded nodes in the VRN. Hence, a node needs to know the load information of other nodes in the VRN. The peer periodically checks the length of the query message queue.

If the proportion of the load change reaches a proportion threshold *load_change*, it notifies the other nodes in the VRN of the load information. We assume that the size of the VRN for a resource r is v and the proportion threshold of the load change for a node n is p_n. In a complete graph, the expected total load updating message in a checking cycle is shown in Eq. (6). We can find that when the size of the VRN is large, the cost of the load information sharing is considerably high.

$$E(\text{cost}) = \sum_{n \in \text{VRN}} p_n \cdot (v - 1). \tag{6}$$

Small-world [14,17] is an important theory and an interesting phenomena in information retrieval. It reveals that two nodes in a power-law distribution network are likely to be connected through a short sequence of intermediate nodes [1]. Based on this theory, we create the power-law links to address the issue of efficiently informing the load change in the VRN.

As proved in [10], the hierarchical networks can be used to model the small-world phenomena. Therefore, to efficiently broadcast the load information in a VRN, we employ an inheritor tree topology instead of the complete graph to organize the VRN, as shown in Fig. 7. In a resource's replica inheritor tree, the original resource node is the root of the tree. Furthermore, any node's parent is the one that creates a new replica on it. For example, node A creates a new replica on node B, and then node B is a child node of A in the inheritor tree. To keep the stability of the replica inheritor tree, it should inform its parent and child nodes when any non-root node leaves the VRN. It is also recommended that each node in the VRN periodically pings its neighbors to update the VRN list.

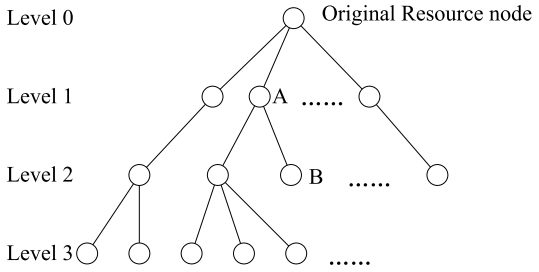


Fig. 7. Replica inheritor tree of a popular resource.

We divide the replica inheritor tree into several levels based on the relationship of replication. Each node belongs to a level as Fig. 7 shows. We define a node's level in the replica inheritor tree as follows. First, the level of the original resource node is level 0. Second, if the level of the parent node of a non-root node n is j, the level of n is $j + 1$. To transfer the load information efficiently, a node creates power-law links to other nodes in the VRN. Based on the replica inheritor tree level, a peer a transfers its load updating messages to another node b by the probability $P(a|b)$ in Eq. (7), where A and C are two constant parameters ($0 < C < 1, A > 1$) and l_a and l_b are node a's and node b's levels, respectively.

$$P(a|b) = C \times A^{-1-|l_a - l_b|}. \tag{7}$$

Using the layered replica inheritor tree, the expected total load updating message in a checking cycle is shown in Eq. (8), where p_n is the proportion threshold of the load change. By using the replica inheritor tree and the power-law links, we greatly reduce the updating messages in the VRN over the complete figure construction. Furthermore, the load information can still be efficiently shared in the nodes of a VRN.

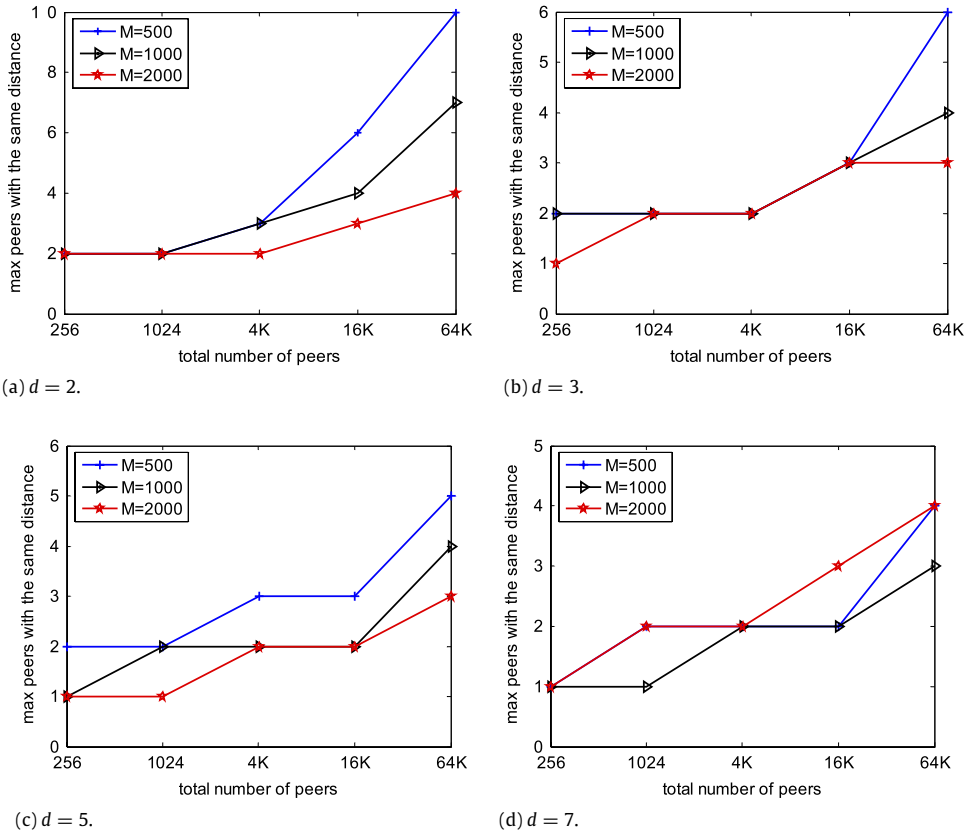
$$E(\text{cost}) = \sum_{n \in \text{VRN}} \sum_{m \in \text{VRN}, m \neq n} (p_n \times C \times A^{-1-|l_n - l_m|}). \tag{8}$$

5. Performance evaluation

5.1. Simulation setup

This section presents the performance evaluation of the FAN network through simulations using PeerSim [31], a P2P simulation framework. All of the experiments are carried out on a Windows PC which has an Intel Pentium 2.8 GHz dual-core CPU and 1 GB main memory.

In the FAN experiments, a peer is described by d-dimensional attributes. Each dimensional attribute is an integer which follows the uniform distribution from 0 to M. Before measuring the statistics, we randomly inject a mixture of operations (peer join, departure and query) into FAN. The proportion of peer join and departure operations is kept roughly equal. Each peer averagely issues 100 queries while online. The FAN routing efficiency is evaluated by measuring the average routing hops in Sections 5.4 and 5.5. We have developed the simulations in Sections 5.2 and 5.3 to illuminate the feasibility and efficiency of FAN subspace management. Furthermore, the simulation in Section 5.6 has been carried out to analyze the process at the super-peers to illustrate the feasibility of the FAN protocol. In the end, we improve the simulations to evaluate the performance of the FAN replication algorithm by measuring the query delay in Section 5.7. Additional parameters in the simulations are shown in Table 4. In this section, we implemented Chord, Mercury and FAN protocols over PeerSim, while the experimental data of CAN comes from Ratnasamy et al. [18].



(a) $d = 2$. (b) $d = 3$.

(c) $d = 5$. (d) $d = 7$.

Fig. 8. Maximal number of peers with the same distance in FAN.

Table 4
Additional parameters in the simulations.

	Parameter descriptions	Values
N	The network size of FAN	256, 1024, 4096, 16 K, 64 K
d	Mapping space dimensions	2, 3, 5, 7
S	Super-peers in a subspace	1
k	The capability of a subspace	4, 8, 12, 16, 20, 24
M	The range of each dimension value	500, 1000, 2000
w	The size of query queue	30
m	Resource amount	100, 1000, 10,000, 100,000

5.2. Maximal number of peers with the same distance

The peer distance is the foundation of subspace partitioning and resource routing in FAN. However, in a d-dimensional Cartesian space, some different peers having different coordinates may have the same peer distance. For example, the peers $P_1(1, 0, 0, 0, 1)$, $P_2(0, 0, 1, 0, 1)$, and $P_3(0, 1, 1, 0, 0)$ have the same distance. Furthermore, according to the subspace management algorithm, the peers in a FAN that have the same distance cannot be placed into different subspaces. Therefore, to keep the consistency of the FAN subspace management, the maximal number of peers with the same distance cannot exceed the subspace capability (i.e. k). We carried out the experiments to evaluate it with various N, d, and M values. In the simulations, every dimensional resource attribute follows the uniform distribution from 0 to M.

The numerical results in Fig. 8 show that, in most situations, the maximal number of peers with the same distance is smaller than 8, i.e. it can be well supported by a subspace. The only exception is when the dimension of resource attributes is small (e.g. $d = 2$) and the total number of peers is very large (more than 64 K). Under such circumstances, we should choose an appropriate M value (bigger than 1000) to reduce the maximal number of peers with the same distance. Therefore, we can conclude that the subspace division strategy in a FAN is feasible and the maximal number of peers with the same distance will not overburden the subspace capability.

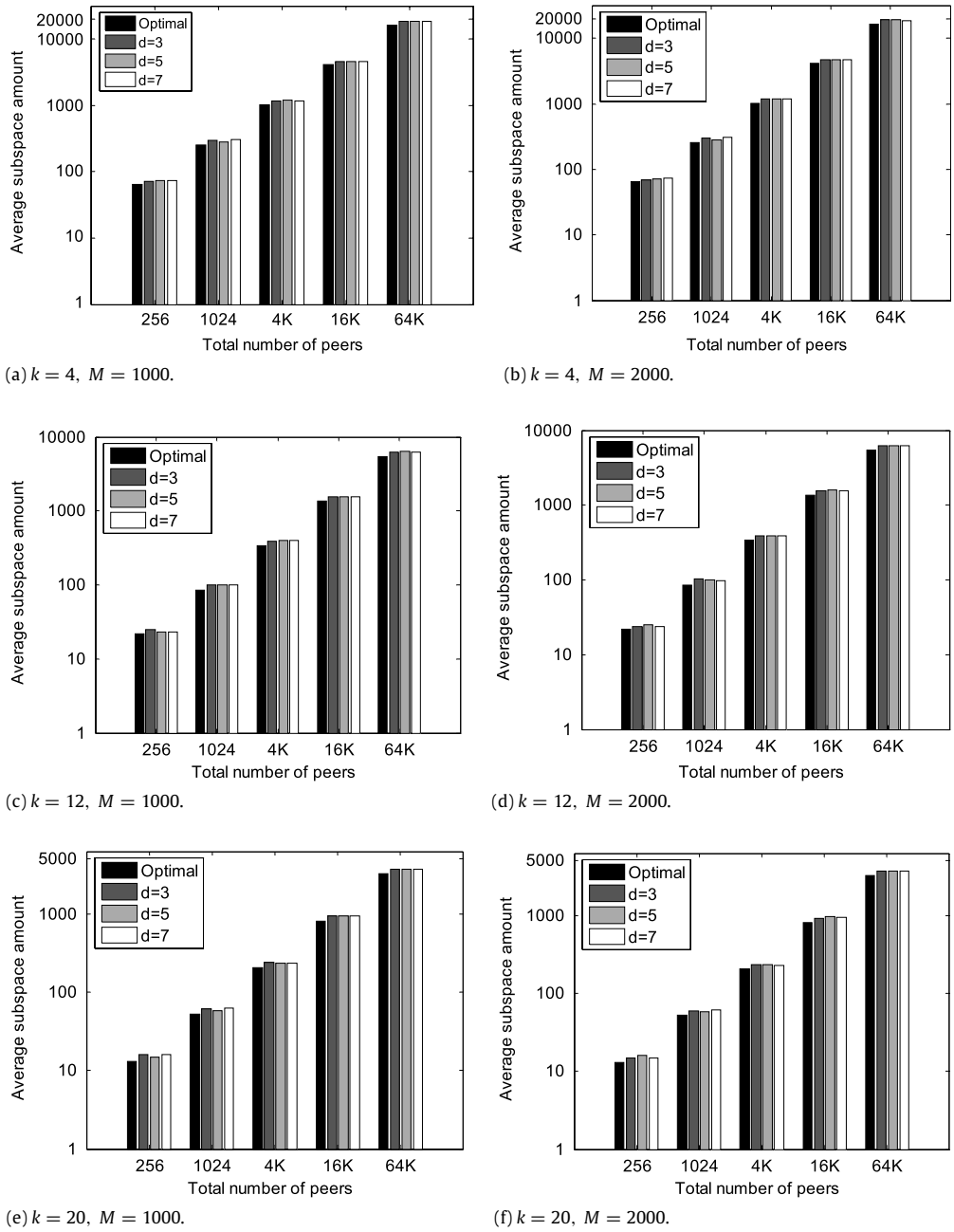
5.3. Average number of subspaces in FAN

As discussed in Section 3, FAN routing efficiency is $O(\log(N/k))$, and the number of messages for peer joining and leaving in a FAN are both $O(\log(N/k)/k)$. Furthermore, both the FAN routing efficiency and the maintenance cost are related to the total number of subspaces. The following analyses are based on the assumption that the total number of subspaces is $O(N/k)$. Therefore, whether the FAN subspace management strategy can efficiently slow the increase of the subspace is crucial for the scalability and availability of FAN. In this simulation, we measured the statistics of average number of subspaces with various N, d, k and M values.

The numerical results in Fig. 9 show that the average number of subspaces is approximately equal to the optimal value (N/k). By analyzing this simulation experiment, we have found that the average number of subspaces has little relation to the d (the dimensions of the resource attributes) and M values, but is mainly related to the k and N values. Therefore, we can draw a conclusion that the FAN subspace management strategy can efficiently keep the number of subspaces as $O(N/k)$. The following experiments and analyses of the FAN resource search and maintenance cost are based on the assumption that the number of subspaces is $O(N/k)$.

5.4. Routing efficiency of the single-dimensional FAN

FAN proposed in this paper is an underlying P2P routing overlay, and the routing efficiency is the most important criterion for an overlay design. FAN uses the extended adjacent subspace links similar to Chord to improve its routing efficiency. Though FAN is a routing algorithm designed to perform efficient resource search



(a) $k = 4,\ M = 1000$. (b) $k = 4,\ M = 2000$.

(c) $k = 12,\ M = 1000$. (d) $k = 12,\ M = 2000$.

(e) $k = 20,\ M = 1000$. (f) $k = 20,\ M = 2000$.

Fig. 9. The average number of subspaces in FAN.

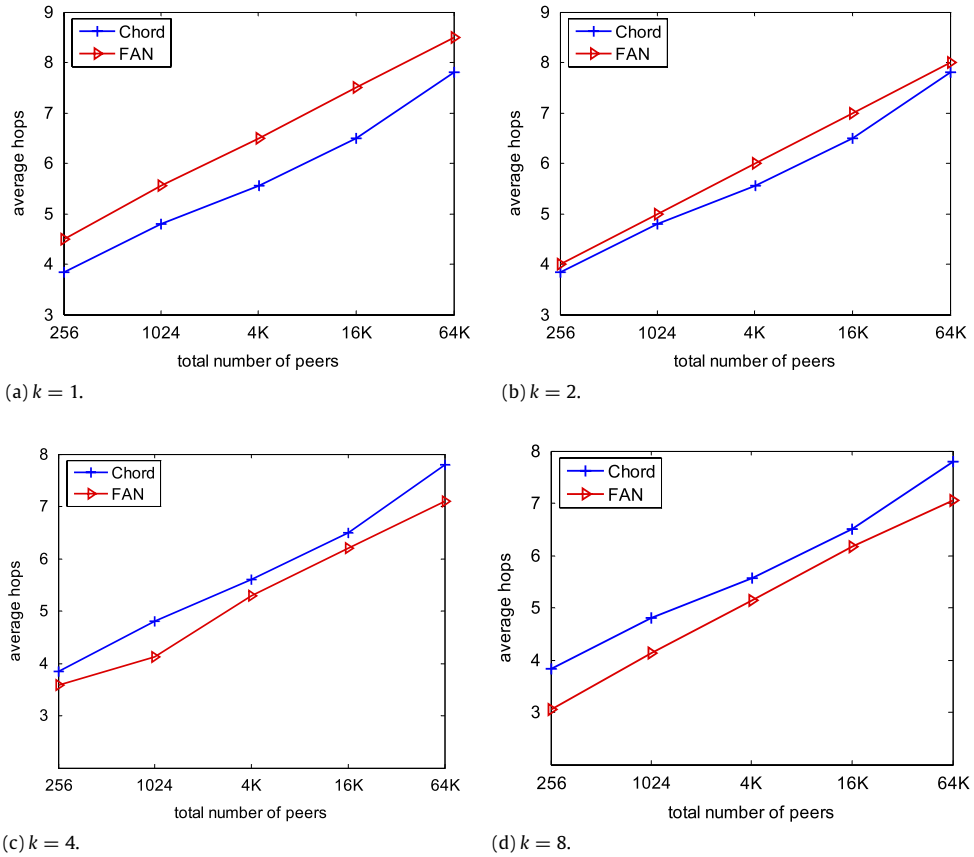
over multi-dimensional attributes, it can also support single-dimensional attribute search. Furthermore, the routing efficiency of a single-dimensional FAN represents the efficiency of the routing message spreading in the links of the extended adjacent subspaces. To evaluate the performance of low-dimensional FAN and the efficiency of message spreading in the links of the extended adjacent subspaces, we designed a simulation experiment to compare the routing efficiency of single-dimensional FAN and Chord.

As the numerical results show in Fig. 10, the single-dimensional FAN can achieve a logarithmic routing efficiency like Chord. As for the smaller k value, e.g. $k = 1, 2$, FAN generates a little bit more hops than Chord. The reason is possibly as follows. The Chord builds a ring topology, and FAN can only transmit routing message in one direction along the extended adjacent subspace links. Therefore, FAN routing efficiency is a little lower than Chord. Nevertheless, for larger k values, FAN shows better performance than Chord. Overall, it is nearly the same as Chord for single-dimensional attribute search. Thus, we can say that FAN is a scalable efficient routing algorithm supporting a single attribute. The routing messages can efficiently spread in the structure of the extended adjacent subspace links.

5.5. Routing efficiency of multi-dimensional FAN

As the main idea described above, FAN is an underlying P2P overlay supporting multi-dimensional resource attributes as CAN. In Section 3, we have theoretically analyzed that the FAN routing efficiency is $O(\log(N/k))$. In this section, we carry out the simulation experiments to evaluate the average routing hops in a multi-dimensional FAN with various N, k, and d values.

The theoretical FAN routing efficiency is $O(\log(N/k))$ that has great relation with the k value and has little relation with the d value. Therefore, we first carry out simulation experiments to



(a) $k = 1$. (b) $k = 2$.

(c) $k = 4$. (d) $k = 8$.

Fig. 10. Routing efficiency comparison of single-dimensional FAN and Chord.

find the relationship between the routing efficiency of FAN and k values. In the experiments, we figure out the average routing hops in FAN with various d, k and N values. As shown in Fig. 11, the FAN routing efficiency gets better with an increasing k value, and FAN works well in the large network size ($N = 64$ K). Furthermore, we also find from the numerical results that the average routing hops have a logarithmic relationship with N/k, which illustrates that our analysis about the FAN routing efficiency is reasonable.

Since FAN can support multi-dimensional attributes, we expect FAN to keep a high routing efficiency over a high attribute dimension. As shown in Fig. 11, the routing efficiency of FAN has little change with various d values. This is because the FAN routing algorithm is based on peer's distance, which has little relation with attribute dimension. The peer's distance is defined as the second moment of the peer to the origin of coordinates, and it contracts multiple attributes into one dimension. Thus, FAN can work well with a large attribute dimension.

CAN and Mercury are both P2P routing frameworks supporting resource search over multi-dimensional attributes. We compare FAN with CAN and Mercury in the same network circumstances. Since CAN uses the greedy forwarding to deliver the routing messages, multiple peers in a zone will be overloaded with the continuous increase in the number of messages. We compare FAN and CAN routing efficiency with $k = 1, 2, 3, 4$ as Ratnasamy et al. [18] suggested. Since FAN employs a super-peer to manage the subspace, we also implement a super-peer over Mercury in the simulation experiments. Furthermore, we have carried out more experiments to investigate how N, k and d values influence the FAN routing efficiency.

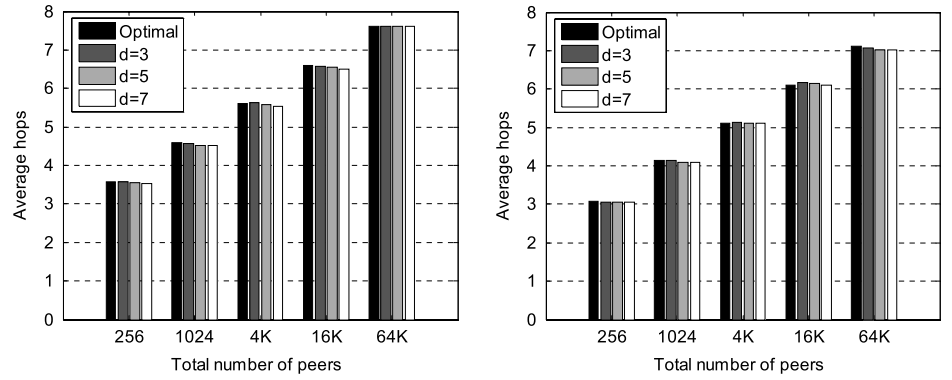
In the second simulation, we compare the routing efficiency of FAN, CAN and Mercury with the same capability k in a subspace. As demonstrated above, FAN has little relation to the resource attribute dimensions. However, the routing efficiency of CAN and Mercury have great relations with the attribute dimensions, whose routing efficiency is $O(dn^{1/d})$ and $O(\log^2 n/k)$, respectively. To simplify the simulation experiments, we compare 3-dimensional FAN, 3-dimensional Mercury, and various d-dimensional CAN. The results in Fig. 12 show that FAN gets a better routing efficiency than CAN and Mercury when their subspaces or zones contain the same maximal number (k) of peers. Furthermore, CAN and Mercury cannot support too many peers in a subspace or zone. Nevertheless, in the experimental results, we find that FAN can support a large k value and achieve a better performance with k growing, which adapts to the actual applications better. Through the experimental results, we can also find that the advantage of FAN routing expands more with increasing network size, and that FAN works well under a huge network size.

Through the simulation experiments in this section, we can draw the conclusion that FAN is an efficient P2P overlay supporting multi-dimensional attributes with $O(\log(N/k))$ routing efficiency.

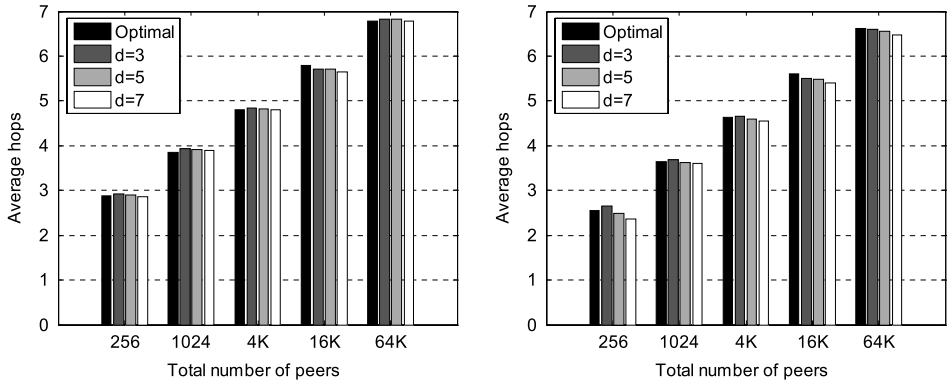
5.6. FAN range query efficiency

FAN also provides an efficient multi-dimensional range query. So, in this section, we design simulation experiments to evaluate FAN range query efficiency. In the experiments, we put $m =$ 100–100,000 resources into the network in which every dimensional attribute is an integer following a uniform distribution form (0, 1000). Each FAN subspace contains $m/(N/k)$ resources at most. We randomly choose a range query region in the mapping space to compare the FAN range query efficiency with VBI-tree [12], Mercury [3], BATON [11], and Squid [19]. Fig. 13 shows the routing messages with various dimension values.

As the experimental results show in Fig. 13, FAN achieves a better range query efficiency while the resource amount increases.

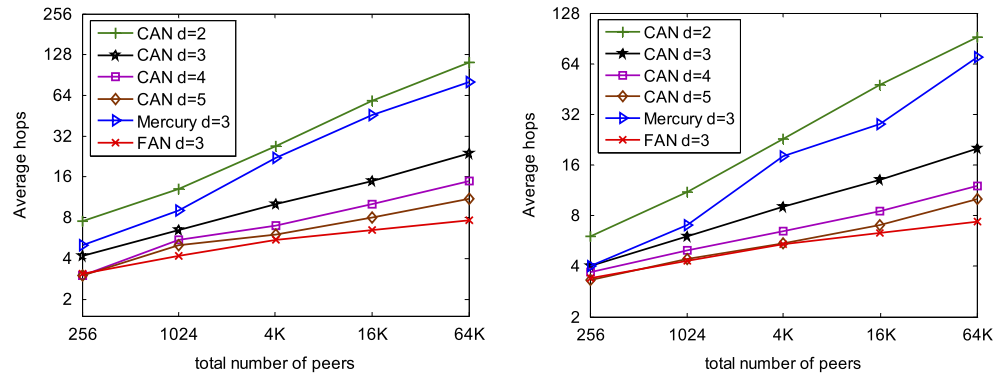


(a) $k = 4$. (b) $k = 8$.

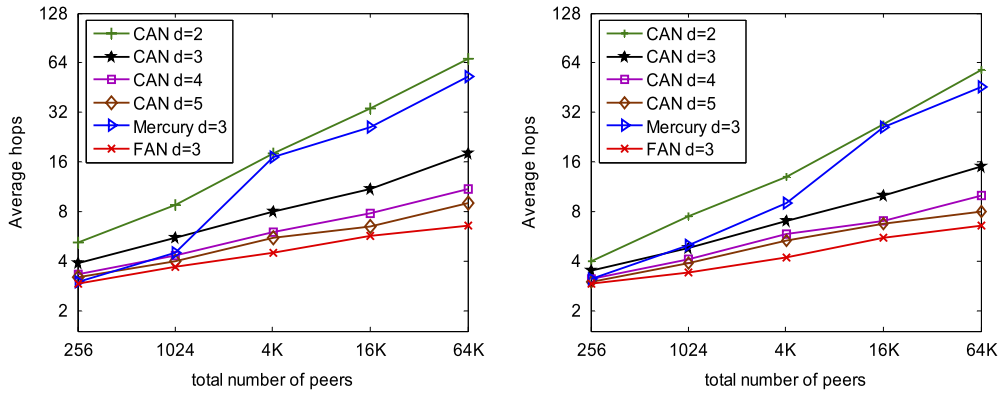


(c) $k = 12$. (d) $k = 16$.

Fig. 11. FAN routing efficiency with various d and k values.

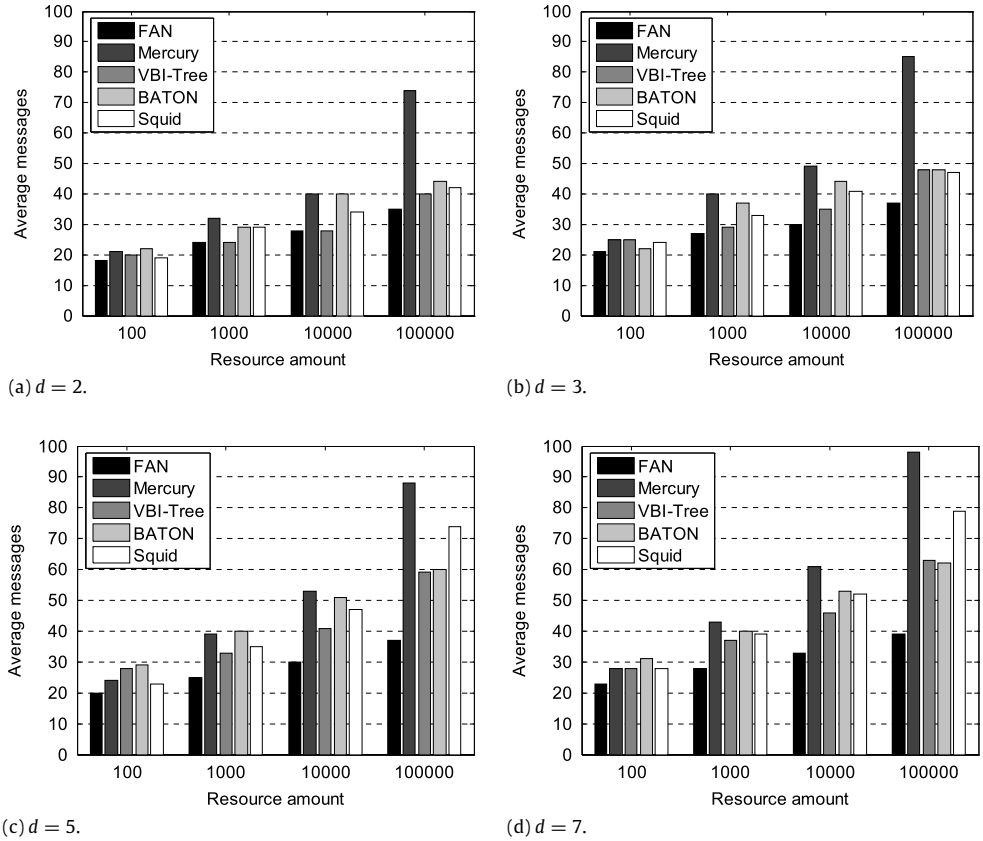


(a) $k = 1$. (b) $k = 2$.



(c) $k = 3$. (d) $k = 4$.

Fig. 12. Routing efficiency comparison of multi-dimensional FAN, CAN and Mercury.



(a) $d = 2$. (b) $d = 3$. (c) $d = 5$. (d) $d = 7$.

Fig. 13. Range query comparison of multi-dimensional FAN, Mercury, VBI-Tree, BATON and Squid.

It also has little effect from the attribute dimensions while the counterparts are even worse with the higher attribute dimensions. As we know, FAN employs resource distance and attribute slope to support range queries. It is obvious that resource distance and attribute slopes are not influenced by the attribute dimension value; that is, FAN scales well with high resource dimensions.

5.7. Routing messages at super-peers in FAN

The links of the extended adjacent subspaces achieve fast resource search in FAN and also make the super-peers process more routing information. Therefore, we should take into account whether the routing processing will overburden the super-peers and influence FAN stability.

We carry out experiments to evaluate the message amount processed at super-peers in FAN with various N, k and d values. In the experiments, we let every peer randomly issue a query message once in a time slot (we use one minute in the experiments). We evaluate the average routing message amount at all super-peers in the period of the time slot.

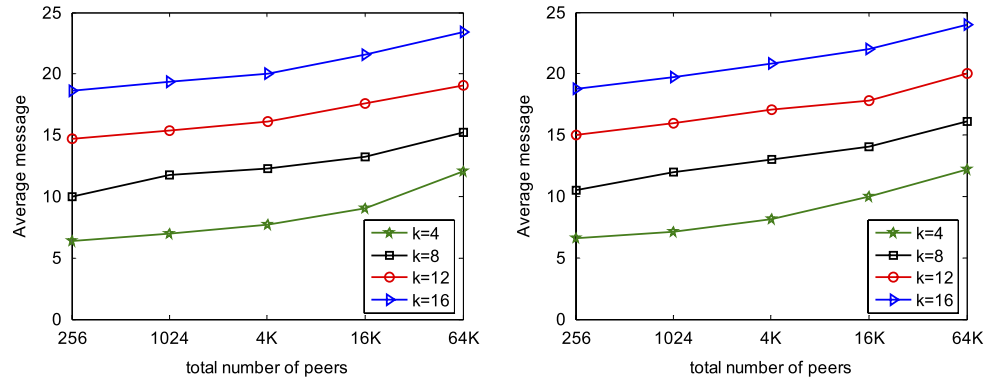
The results in Fig. 14 show that the routing messages processed at super-peers get larger with increasing k value. The results also show that the routing messages processed at super-peers have a logarithmic relationship with the total number of peers N. Furthermore, the average amount of routing messages at super-peers is acceptable (less than 30), even though the total number of peers in FAN is more than 64 K. Hence, processing the routing messages will not overburden the FAN super-peers. Considering the heterogeneous capacities of nodes, this work takes advantage of the heterogeneity feature to achieve better performance.

5.8. Query delay in FAN

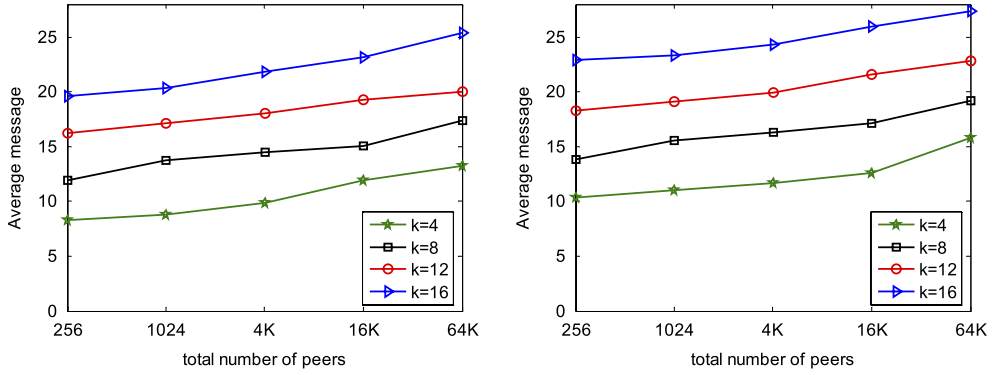
The simulations in Sections 5.4 and 5.5 have demonstrated the FAN routing efficiency by evaluating the average routing hops. Furthermore, the query delay also has a great effect on the routing efficiency of a network topology. FAN employs the adaptive popular resource replication to relieve the hot nodes and decrease the query delay. Hence, we carried out the simulation experiments to evaluate the contributions of the adaptive replication on the query delay. We compare the FAN average query delay in two cases: with the adaptive replication and without the adaptive replication.

In the simulations, we make each node keep a query queue whose size is 30. When the query queue at the experimental node in FAN with the replication algorithm has reached half full, the nodes enter flow control mode to balance the load. However, the query queue at the nodes in FAN without adaptive replication is considerably long. In both cases, a node processes the received queries in the FIFO mode. To simulate the imbalance of the resource distribution and access, we assume that the processing time for each query is 10 ms, and both the resource distribution and access follow the power-law distribution of exponent 3.0. The other experimental parameters are $load\_change = 0.1$, $A = 2$ and $C = 0.5$ in Eq. (7). All the simulations ignore the network transmission delay.

We design the simulations to evaluate the FAN query delay under various network loads. The experimental results are shown in Figs. 15 and 16. FAN can work well by keeping a low query delay under a light network load whether it employs the adaptive replication or not (see Fig. 15). However, the adaptive replication is a great benefit to the query delay of FAN with a huge network

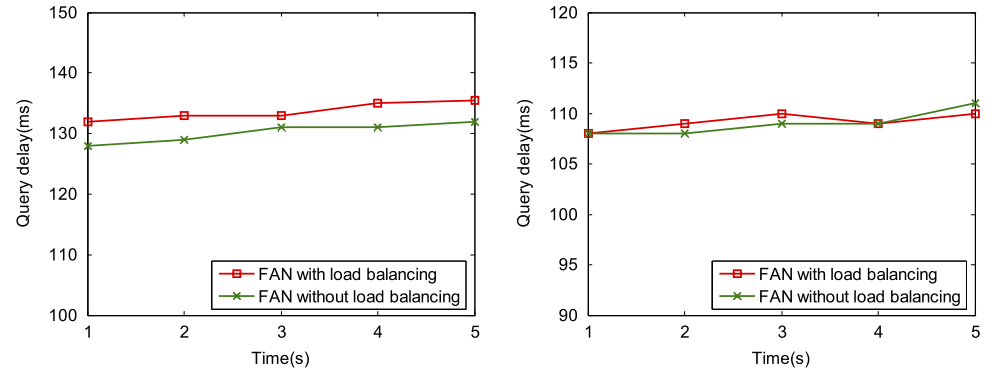


(a) $d = 2$. (b) $d = 3$.

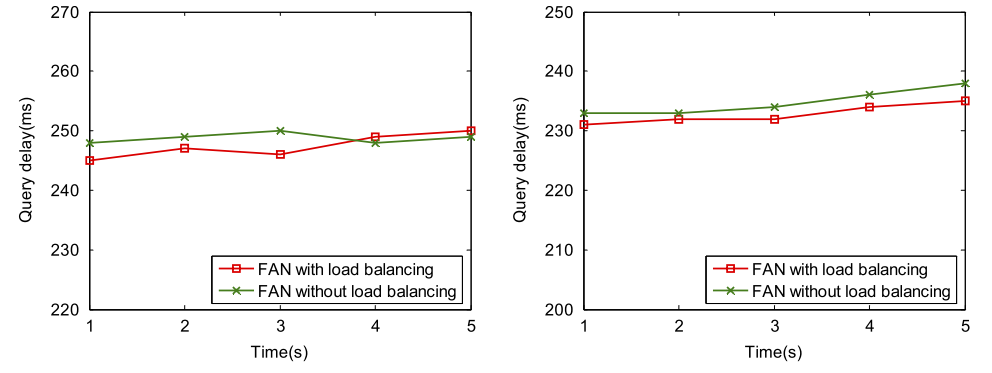


(c) $d = 5$. (d) $d = 7$.

Fig. 14. The routing messages at super-peers in FAN.

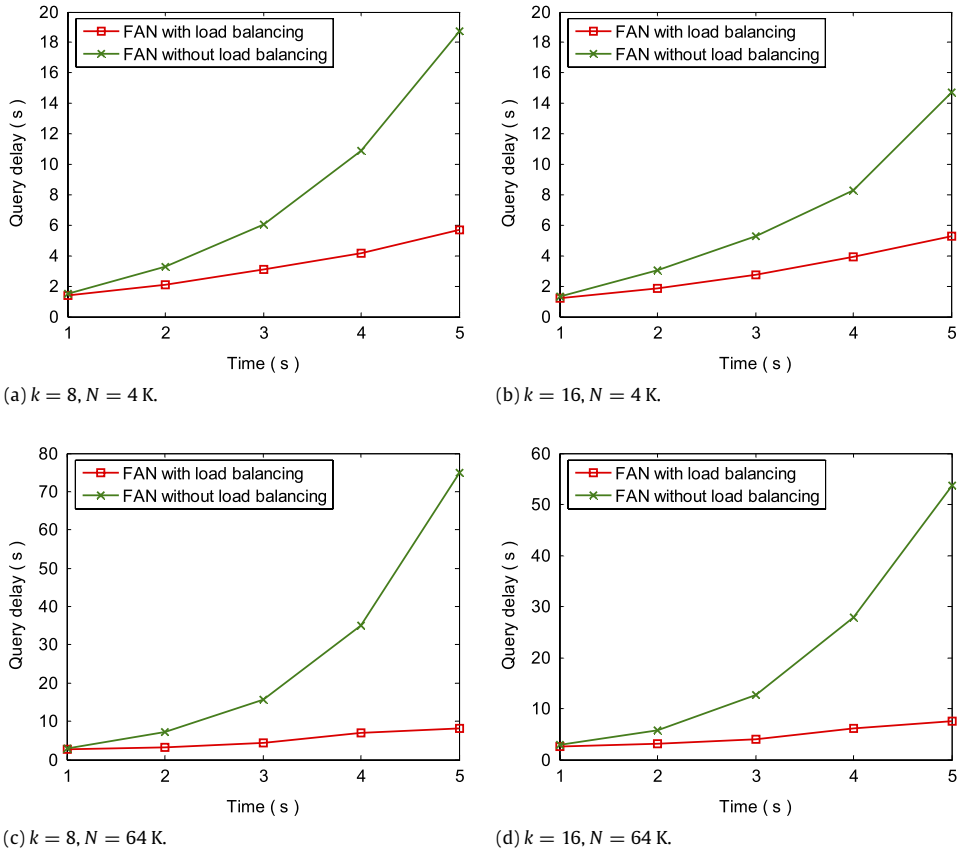


(a) $k = 8, N = 4$ K. (b) $k = 16, N = 4$ K.



(c) $k = 8, N = 64$ K. (d) $k = 16, N = 64$ K.

Fig. 15. Query delay experiments over the light network load (each node issues only one query in a second).



(a) $k = 8, N = 4$ K. (b) $k = 16, N = 4$ K.

(c) $k = 8, N = 64$ K. (d) $k = 16, N = 64$ K.

Fig. 16. Query delay experiments over the heavy network load (each node issues 10 queries in a second).

load (see Fig. 16). Without the load balancing approach, the query queue at each node increases sharply when the number of queries rises rapidly. The query has to wait a long time at the FIFO query queues of the hot nodes. As a consequence, the query delay reaches an unacceptable level. However, if we use the adaptive replication algorithm to balance the load of hot nodes, most of the queries are satisfied on a lower number of hops by efficiently creating replicas for popular resources on the hot accessing path. Thus, the adaptive replication algorithm can efficiently alleviate the query delay of FAN.

6. Conclusion

Most of the existing work on P2P networks does not support content search over multiple resource attributes. To address this issue, this paper proposes the FAN network as an underlying P2P overlay supporting efficient resource search over multi-dimensional attributes. As the analysis and experimental results show, FAN has advantages in routing efficiency and maintenance cost over the current search algorithms. Through the experiments, we demonstrate that FAN is suitable even in a network environment with fast peer joining and leaving, very large network size, and high dimension of resource attributes. This paper also presents a range query algorithm over FAN based on resource distance and attribute slope. It can achieve an efficient and scalable range query, especially when the resource amount increases and the attribute dimension grows higher. Furthermore, this paper presents an adaptive replication algorithm over FAN through the use of a virtual replica network (VRN). It can significantly reduce the querying delay for a FAN under a heavy network load by creating replicas of the popular resources and forwarding the queries to the lightly loaded nodes. For future work, the optimization of FAN towards reducing the cost based on various parameters will be further facilitated. Furthermore, a P2P resource sharing system supporting multi-attribute search based on FAN can be implemented and tested over PlanetLab to evaluate the performance of the algorithms in real applications.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grants 60873225, 60773191, and 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, Natural Science Foundation of Hubei Province under Grant 2009CDB298, Wuhan Youth Science and Technology Chenguang Program under Grant 200950431171, Open Foundation of State Key Laboratory of Software Engineering under Grant SKLSE20080718, Innovation Fund of Huazhong University of Science and Technology under Grants 2010MS068 and Q2009021, Fundamental Research Fund of Wuhan University under Grant 6082024 and US National Science Foundation Grants CNS-0834592, CNS-0832109 and CNS-0953909. The authors would like to thank the anonymous reviewers for their valuable comments.

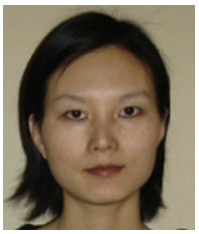
References

[1] R. Albert, H. Jeong, A.-L. Barabási, Diameter of the world wide web, Nature 401 (1999) 130–131.
[2] M.A. Arefin, M.Y.S. Uddin, I. Gupta, et al., Q-tree: a multi-attribute based range query solution for tele-immersive framework, in: Proc. of the 29th International Conference on Distributed Computing Systems, ICDCS, 2009, pp. 299–307.
[3] A.R. Bharambe, M. Agrawal, S. Seshan, Mercury: supporting scalable multi-attribute range queries, in: Proc. of the 2004 ACM SIGCOMM Conference, 2004, pp. 353–366.
[4] I. Bhattacharya, S.R. Kashyap, S. Parthasarathy, Similarity searching in peer-to-peer databases, in: Proc. of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS, 2005, pp. 329–338.

[5] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, et al., A case study in building layered DHT applications, in: Proc. of the 2005 ACM SIGCOMM Conference, 2005, pp. 97–108.
[6] A. Crainiceanu, P. Linga, A. Machanavajjhala, *P*-ring: an efficient and robust P2P range index structure, in: Proc. of the 2007 ACM SIGMOD Conference, 2007, pp. 223–234.
[7] M. Demirbas, H. Ferhatosmanoglu, Peer-to-peer spatial queries in sensor networks, in: Proc. of the 3rd IEEE International Conference on Peer-to-Peer Computing, P2P, 2003, pp. 32–39.
[8] P. Ganesan, M. Bawa, H. Garcia-Molina, Online balancing of range-partitioned data with applications to peer-to-peer systems, in: Proc. of the 30th International Conference on Very Large Data Bases, VLDB, 2004, pp. 444–455.
[9] J. Gao, P. Steenkiste, An adaptive protocol for efficient support of range queries in DHT-based systems, in: Proc. of the International Conference on Network Protocols, ICNP, 2004, pp. 239–250.
[10] J. Gray, P. Helland, P. O'Neil, et al., The dangers of replication and a solution, in: Proc. of the 1996 ACM SIGMOD Conference, 1996, pp. 173–182.
[11] H.V. Jagadish, B.C. Ooi, Q.H. Vu, BATON: a balanced tree structure for peer-to-peer networks, in: Proc. of the 31st International Conference on Very Large Data Bases, VLDB, 2005, pp. 661–672.
[12] H.V. Jagadish, B.C. Ooi, Q.H. Vu, et al., VBI-tree: a peer-to-peer framework for supporting multi-dimensional indexing schemas, in: Proc. of the 22nd International Conference on Data Engineering, ICDE, 2006, p. 34.
[13] D.R. Karger, E. Lehman, F. Leighton, et al., Consistent hashing and random trees: distributed cashing protocols for relieving hot spots on the world wide web, in: Proc. of the 29th Annual ACM Symposium Theory of Computing, 1997, pp. 654–663.
[14] J. Kleinberg, Small-world phenomena and the dynamics of information, in: Advanced in Neural Information Processing Systems (NIPS), vol. 14, The MIT Press, 2001, pp. 431–438.
[15] B. Liu, W.-C. Lee, D.L. Lee, Supporting complex multi-dimensional queries in P2P systems, in: Proc. of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS, 2005, pp. 155–164.
[16] T. Locher, S. Schmid, R. Wattenhofer, eQuus: a provably robust and locality-aware peer-to-peer system, in: Proc. of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P), Cambridge, United Kingdom, 2006, pp. 3–11.
[17] S. Milgram, The small world problem, Psychology Today 2 (1967) 60–67.
[18] S. Ratnasamy, P. Francis, M. Handley, et al., A scalable content-addressable network, in: Proc. of the 2001 ACM SIGCOMM Conference, 2001, pp. 161–172.
[19] C. Schmidt, M. Parashara, Squid: enabling search in DHT-based systems, Journal of Parallel and Distributed Computing 68 (7) (2008) 962–975.
[20] H. Shen, C.-Z. Xu, Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks, IEEE Transactions on Parallel and Distributed Systems 18 (6) (2007) 849–862.
[21] Y.F. Shu, B.C. Ooi, K.L. Tan, et al., Supporting multi-dimensional range queries in peer-to-peer systems, in: Proc. of the Fifth IEEE International Conference on Peer-to-Peer Computing, P2P, 2005, pp. 173–180.
[22] W. Song, R. Li, Z. Lu, et al., FAN: a flabellate overlay network supporting scalable multi-dimensional attributes, in: Proc. of the IEEE 22nd International Conference on Advanced Information Networking and Applications, AINA, Okinawa, Japan, 2008, pp. 1005–1012.
[23] I. Stoica, R. Morris, D. Karger, et al., Chord: a scalable peer-to-peer lookup service for internet applications, in: Proc. of the 2001 ACM SIGCOMM Conference, 2001, pp. 149–160.
[24] C. Tang, Z. Xu, M. Mahalingam, pSearch: information retrieval in structured overlays, ACM SIGCOMM Computer Communications Review 33 (1) (2003) 89–94.
[25] Y. Tang, J. Xu, S. Zhou, et al., *m*-light: indexing multi-dimensional data over DHTs, in: Proc. of the 29th International Conference on Distributed Computing Systems, ICDCS, 2009, pp. 191–198.
[26] C. Wang, B.A. Alqaralleh, B.B. Zhou, et al., Self-organizing content distribution in a data indexed DHT network, in: Proc. of the 6th International Conference on Peer-to-Peer Computing, P2P, 2006, pp. 241–248.
[27] B. Yang, H. Garcia-Molina, Design a super-peer network, in: Proc. of the 19th International Conference on Data Engineering, ICDE, 2003, pp. 49–60.
[28] B.Y. Zhao, L. Huang, J. Stribling, et al., Tapestry: a resilient global-scale overlay for service deployment, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 41–52.
[29] C. Zheng, G. Shen, S. Li, et al., Distributed segment tree: support of range query and cover query over DHT, in: Proc. of 5th International Workshop on Peer-to-Peer Systems, IPTPS, 2006.
[30] Y. Zhu, Y. Hu, Towards efficient load balancing in structured P2P systems, in: Proc. of the 18th International Parallel and Distributed Processing Symposium, IPDPS, 2004, p. 20.
[31] PeerSim. http://sourceforge.net/projects/PeerSim.

Ruixuan Li received the B.S., M.S. and Ph.D. in Computer Science from Huazhong University of Science and Technology, China in 1997, 2000 and 2004, respectively. He is currently an Associate Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology and a Visiting Researcher in Department of Electrical and Computer Engineering at University of Toronto. His research interests include peer-to-peer computing, distributed data management, and distributed system security. He is a member of IEEE and ACM.

Wei Song received the B.S. degree in Mechanical Science and Engineering from Huazhong University of Science and Technology, China in 2001. He is currently a Ph.D. candidate in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include peer-to-peer network, distributed system, and distributed system security.



Haiying Shen received the B.S. degree in Computer Science and Engineering from Tongji University, China in 2000, and the M.S. and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Holcombe Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed and parallel computer systems and computer networks, with an emphasis on peer-to-peer and content delivery networks, wireless networks, resource management in cluster and grid computing, and data searching. She is a member of IEEE and ACM.

Weijun Xiao received the B.S. and M.S. degrees in Computer Science from Huazhong University of Science and Technology, China in 1995 and 1998, respectively, and the Ph.D. degree in Computer Engineering from University of Rhode Island in 2009. He is currently a postdoctoral associate in the Department of Electrical and Computer Engineering at University of Minnesota. His research interests include computer architecture, networked storage system, embedded system, and performance evaluation. He is a member of IEEE and IEEE Computer Society.

Zhengding Lu received the B.S. degree in Mathematics from Wuhan University, China, in 1967, and the M.S. degree in Computer Science and Engineering from Chinese Academy of Sciences in 1982. He is currently a full Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology, China. His research interests include distributed computing, database systems, information security, and performance optimization. He is a member of IEEE and IEEE Computer Society, and the Director of China Computer Federation.