Front. Electr. Electron. Eng. China 2010, 5(1): 49–58
DOI 10.1007/s11460-009-0068-0

RESEARCH ARTICLE

Haiping HUANG, Yan ZHENG, Hongwei CHEN, Ruchuan WANG

PChord: a distributed hash table for P2P network

Abstract As a solution for data storage and information sharing for peer-to-peer (P2P) networks, a novel distributed hash table (DHT) structure called PChord is presented in this paper. PChord adopts a bi-directional searching mechanism superior to Chord and enhances the structure of the finger table. Based on Hilbert space filling curve, PChord realizes the mapping mechanism for multi-keyword approximate searching. Compared with the Chord and Kademlia protocols, PChord evidently increases speed on resource searching and message spreading via theoretic proof and simulation results, while maintaining satisfactory load balance.

Keywords distributed hash table (DHT), peer-to-peer (P2P), bi-directional searching

1 Introduction

The peer-to-peer (P2P) system has the properties of non-central, self-organized, redundant storage, most symmetrical connections, etc. [1]. P2P file sharing systems are characterized by highly replicated content distributed among nodes with enormous aggregate resources for storage and communication [2]. Despite the variety of features, the critical issue in most P2P file systems is efficient location and searching of data items. Distributed hash table (DHT) appropriately provides functions for resource location and data lookup in a dynamic P2P file system [3,4]. A DHT is actually a huge hash table maintained by large numbers of peer nodes in global scope. The DHT application system would map an accessing resource to a unique node identification within global space via secure hash algorithm (SHA), and then find one specific node using distributed searching protocol according to a given identification [5]. Among these methods, DHT is popular in P2P systems because of better scalability, resources load balancing, as well as avoiding the performance bottleneck caused by centralized server mode. Recently, many protocols based on DHT have been proposed such as Chord [6], content-addressable network (CAN) [7], Pastry [8], Tapestry [9], Kademlia [10], and Kelips [11].

CAN adopts a d-dimensional (dD) Cartesian coordinate space (for some fixed d) to implement a DHT targeting the compromise between node degree and network diameter. However, compared to Chord, CAN has lower lookup efficiency because of its need for global information and requires an additional maintenance algorithm to periodically remap the identifier space onto nodes. Different from Chord-like protocols, Pastry is a prefix-based routing protocol, which limits its application. Tapestry is the most similar protocol to Chord. However, it also needs the support of global information and would lead to load imbalance. The Kademlia application system holds two DHTs, namely keywords dictionary and document index dictionary, but long response time becomes its defect especially in large-scale P2P networks.

Chord protocol is proposed to minimize the path length and latency and has the merits of load balance, scalability, availability, and flexible naming [12,13]; whereas, Chord protocol still could be enhanced on aspects of searching speed and routing performance. In order to improve the Chord DHT structure, this paper puts forward a new DHT structure—PChord.

The rest of this paper is organized as follows: in Sect. 2, we review the basic knowledge about Chord. Section 3 presents the details on the improved structure of PChord and realization of multi-keywords approximate searching. Section 4 provides performance analysis on searching time and message spread speed. Section 5 shows the experimental conditions and simulation results. Finally, Sect. 6 concludes this paper.

Received May 21, 2009; accepted August 11, 2009

Haiping HUANG, Yan ZHENG, Hongwei CHEN,

Ruchuan WANG (✉)

College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
E-mail: wangrc@njupt.edu.cn

2 Overview about Chord protocol

The Chord protocol provides fast distributed computation of a hash function mapping keys to nodes using consistent hashing method [6,14]. Consistent hashing has several excellent characteristics. First, the hash function can balance load, that is, all the nodes can receive nearly the same amount of keywords. In addition, if there are N nodes in the system, when the last node joins or leaves the system, only $1/N$ keywords are required to be moved to other locations. In Chord, the nodes only need to know the information of some relevant nodes. The consistent hash function assigns each node and key an m-bit identifier using a base hash function such as SHA-1. The length m of the identifier must be long enough in order to guarantee that the probability of any two nodes or keywords receiving the same identifier is negligibly small. The identifier of a node can be the hash of its Internet protocol (IP) address, while the identifier of the keyword can be the hash value of this keyword itself.

In consistent hashing, each keyword is stored in its successor node. A successor node is the first contiguous node whose identifier is greater than or equal to keyword K identifier, and can be denoted as successor(K). As shown in Fig. 1(a), identifiers are ordered in an identifier circle modulo 2^m, and here the value of m is 6, i.e., there exist up to more than 64 node identifiers along clockwise in this circle. For example, assuming that a node with the IP address 202.119.230.8 would get an identifier 54 through the hash calculation and its successor node is N_{56}. In the same way, successor(10) is similarly available corresponding to N_{14}. Another feature of consistent hashing is that the impact to Chord can be minimized when nodes join or leave. When a node joins the network, in order to maintain consistent hashing mapping, some keywords originally distributed to the successor of node x will be assigned to the new comer. When a node leaves the network, all keywords assigned to it will be re-allocated to the successor of the node clockwise from it.

As mentioned above, let m be the number of bits in the keyword/node identifier. Each node, for instance, node x, maintains a routing table with (at most) m entries, called the finger table described in Fig. 1.

In the global space of 2^m, the finger table of each node has m items (from item 1 to item m) in Chord protocol, expressed as

$$\begin{aligned} x.\text{finger}[i] &= (x + 2^{i-1}) \bmod 2^m, \\ s &= \text{successor}(x.\text{finger}[i]),\ 1 \leqslant i \leqslant m. \end{aligned} \tag{1}$$

The ith entry in the table of node x includes the identity of the first node, s, that succeeds node x by at least 2^{i-1} on the identifier circle, denoted by successor(x.finger[i]).

For example, for N_8 ($x = 8$) in Fig. 1(a), its finger table contains the following node identifiers: N_{14}, N_{21}, N_{32} and N_{42}.

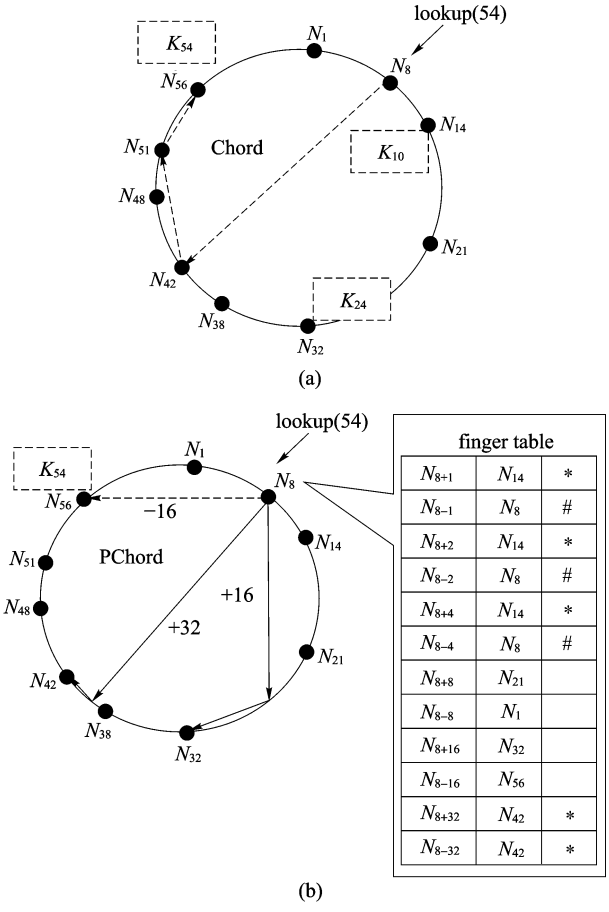

Fig. 1 Searching mode compared between Chord and PChord. (a) Chord consistent hash; (b) PChord data organization

This mechanism has two obvious features. First, each node only needs to know the information of some related nodes. The nearer a node is, the more information is handled. Second, the finger table of each node does not contain all the keywords' locations. For example, in Fig. 1(a), N_8 does not know the location of K_{54}, which is excluded from the finger table of N_8.

In a system composed by N nodes, each node only has to maintain the information of other $O(\log N)$ nodes. In the same way, only $O(\log N)$ messages are in demand each time. When a node joins or leaves the system, Chord has to update the routing information, $O(\log^2 N)$ messages are transferred. (In this paper, $\log x$ is used to mean $\log_2 x$)

3 PChord protocol

3.1 Improved searching structure

PChord protocol uses a variant of the calculating method of consistent hashing to distribute keywords to nodes,

which inherits all merits from Chord simultaneously developing and enriching on resource searching and information dissemination.

If the identifier is an m-bit binary value, there are at most 2^m nodes on the PChord DHT circle, then successor(K) is the nearest node of the keyword K in clockwise direction. For example, in Fig. 1(b), the value of m is 8, and N_{14} = successor(10).

As mentioned above, each node only maintains a finger table (or routing table) with m items at most in Chord. PChord exchanges for searching time efficiency with a little expense of storage space. The finger table of PChord needs $2m$ entries in the space of 2^m, as shown in Fig. 1(b). Therefore, we could enhance the structure of Chord and deduce the following equation in order to describe the routing calculation in PChord:

$$x.\text{finger}[i] = \begin{cases} x + 2^{(i-1)/2}, & i = 2r-1,\ 1 \leqslant r \leqslant m, \\ x - 2^{i/2}, & i = 2r,\ 0 \leqslant r \leqslant m. \end{cases} \quad (2)$$

Chord can only search along a single direction, while PChord realizes the bidirectional lookup to improve searching efficiency. The searching efficiency is greatly improved although the storage of the PChord finger table is doubled. As shown in Fig. 1(a), in Chord searching, N_8 needs to find the successor of K_{54}. Because the farthest pointer of N_8 is N_{42}, N_8 requires N_{42} to further quest the successor of K_{54}, in fact, N_{56}. Similarly, after through N_{51}, N_{56} would look up its finger table and find that the successor to K_{54} is itself, and the whole searching routing process could be depicted by dashed arrows in Fig. 1(a). Figure 1(b) shows that in PChord we only need one hop to find N_{56} via its finger table.

This protocol would be enhanced further, which is only concerned about the efficiency of external keywords searching on the DHT circle and neglects time complexity within the finger table, even if only $O(m)$ complexity is in the worse situation of the internal finger searching ($O(2^m)$ complexity is in that of external keywords searching on the DHT circle). More detailed interpretations are proposed as follows.

Seeing the finger table from Fig. 1(b), we can discover some iterant items which are marked with the symbol "*" such as N_{14} or N_{42}, and some redundant entries on behalf of themselves which are labeled with the symbol "#" such as N_8, all of which should be deleted from the finger table so as to increase the efficiency. Binary searching algorithm would be executed with bi-directions. As shown in Fig. 2, for instance 2^6 nodes in DHT hash space, we demonstrate the process as follows.

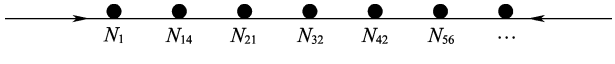

Fig. 2 Improved finger table structure in PChord

Step 1 Calculate $N[i]$= x.finger[i] for each value of i, i = 0,1,...,$2m-1$.

Step 2 Insert $N[i]$ to PChord node's finger table with incremental order excluding the following two situations:

1) The same value as $N[i]$ exists.

2) $N[i]$ is just as the identifier of the current node.

Step 3 Binary searching could be executed in ordered finger table after repeating Steps 1 and 2.

The improved PChord finger table structure obtains $O(\log m)$ performance using binary searching algorithm substituting for original $O(m)$, and contemporarily saves more storage space through eliminating redundant entries.

3.2 Realization of multi-keywords approximate searching

There still exists an intractable hardship in the Chord or PChord protocol which only supports single keyword precise query instead of multi-keywords approximate query. This means searching failure once there is no identical identifier on the DHT circle. In practice, approximate resources would be needed for applications in case of the absence of an accurate keyword [15]. We propose a multi-keyword approximate searching mechanism as subsidiary to PChord mentioned above.

First, some related concepts are given as follows [16].

Definition 1 Characteristic array (CA). By using a consistent hash function, a sequence of attribute keywords denoted by $\{K_1,K_2,...,K_d\}$ for describing the characters of data (or resource) items in PChord would be mapped to an array of $\{W_j | 1 \leqslant j \leqslant d\}$ named CA, where W_j = hash(K_j) and d denotes the number of dimensions.

Definition 2 Hilbert space filling curve (HSFC). Given d-dimensional space denoted by $\mathbb{R}^d$, HSFC realizes the continuous mapping between $\mathbb{R}^d$ and one-dimensional space I, labeled by $H_t^d : \mathbb{R}^d \to I$. The value in I is called Hilbert code, such as "1100" which uses prefix coding algorithm and symbol t represents hierarchy value. If attribute node x in the PChord hash space (i.e., one-dimensional space I) belongs to $\mathbb{R}^d$, then $H(x) \in I$. Two-dimensional (2D) and three-dimensional (3D) HSFCs are shown in Fig. 3, constructed by repeating bi-fractal cutting onto each sub-cube produced recursively along the median line. In general, the total times of recursion equals the value of t, especially in $\mathbb{R}^d$, the length of Hilbert code is dt.

Definition 3 Characteristic index space (CIS). In P2P system, each data item can be demonstrated by a CA, and all data items form the CIS. Based on CIS, using HSFC, each data item denoted by a_i=$\{W_{i1},W_{i2},...,W_{id}\}$, i = 1,2,..., N, would be mapped into the PChord one-dimensional (1D) hash space, corresponding to identifier N_i (for example N_8).

Definition 4 Array similar degree (ASD). Given one array a_i=$\{W_{i1},W_{i2},...,W_{id}\}$, we could define its characteristic value $\alpha_{\lambda_{ij}} = C_{W_j} \lg(N/C_{D_j})$, j = 1,2,...,d, where C_{W_j} denotes the frequency that W_{ij} appears in a_i and C_{D_j}

denotes the total number of data items containing W_{ij}. Given the other array $q_i = \{y_{i1},y_{i2},...,y_{id}\}$, we provide the ASD between a_i and q_i expressed by

$$S(a_i,q_i) = \frac{a_i q_i}{|a_i||q_i|} = \frac{\sum_{j=1}^{d} a_{\lambda_{ij}} y_{ij}}{\left[\sum_{j=1}^{d} a_{\lambda_{ij}}^2 \sum_{j=1}^{d} y_{ij}^2\right]^{1/2}},\ i = 1,2,...,N. \quad (3)$$

We use 3D HSFC to construct CIS in PChord, where hierarchy value t depends on the value m (2^m nodes on the DHT circle) and the number of dimension, and dimension value d determines the keyword counts in each data item. As shown in Fig. 3,

$$H_t^d : \mathbb{R}^d \to 2^m,\ a_i = \{W_{i1},W_{i2},W_{i3}\},$$
$$d = 3,\ t = 2,\ m = 6,\ i = 1,\ 2,...,64. \quad (4)$$

Equation (4) would be explained as using 3D HSFC with 6-bit code length to construct CIS, which should be mapped into 2^6 space. According to Definition 4, multi-keyword query requirement should be represented by the array q_i, and we could realize the approximate searching in PChord by the calculation of $S(a_i,q_i)$, where a_i is the data item in CIS based on HSFC.

Different data items which are in the same line just like a_4, a_5, and a_{12} along the W_2 coordinate direction, might have one common keyword. Also, a_6 is the neighbor item of a_3, a_5, and a_7, which has the same keyword with them in respectively distinct directions. The more the same keywords that two data items have, the closer both coordinates are. Therefore, besides the basic finger table, each one of the data items would also record their neighbor's identifier after mapping for supporting multiple keywords query and approximate query.

Now we give the whole query process in PChord as follows.

Step 1 Suppose a query is sent from a PChord node N, it would search the matching HSFC code length in the finger table of node N.

Step 2 PChord uses a bi-direction finger table to start, which could make the query more efficient as described above.

Step 3 If the matching length is found in the finger table, it would send the request directly to the corresponding node and get all related data items on each of the coordinate axis as the approximate results of the query; else if not, it would select the best related one such as the nearest neighbor whose code length is less than that of the node in its finger table, and the selected node would continue taking on the query task.

The steps mentioned above do not increase the searching complexity still as $O(\log N^{2/3})$. The experiments about searching validity between Chord where approximate searching is not adopted and PChord will be offered in Sect. 5.3.

4 Performance analysis on searching time and message spread

4.1 Analysis on path length

Figure 4 is the comparison on path length between Chord and PChord (hash space is 2^4). Searching length refers to the number of hops when a node finds another in hash space, while average searching length refers to the average number of hops a node finds any node in hash space. As seen in Fig. 4, PChord is more efficient than Chord, especially at the left part of the DHT circle.

Another routing representation about PChord and Chord is found as follows [5]:

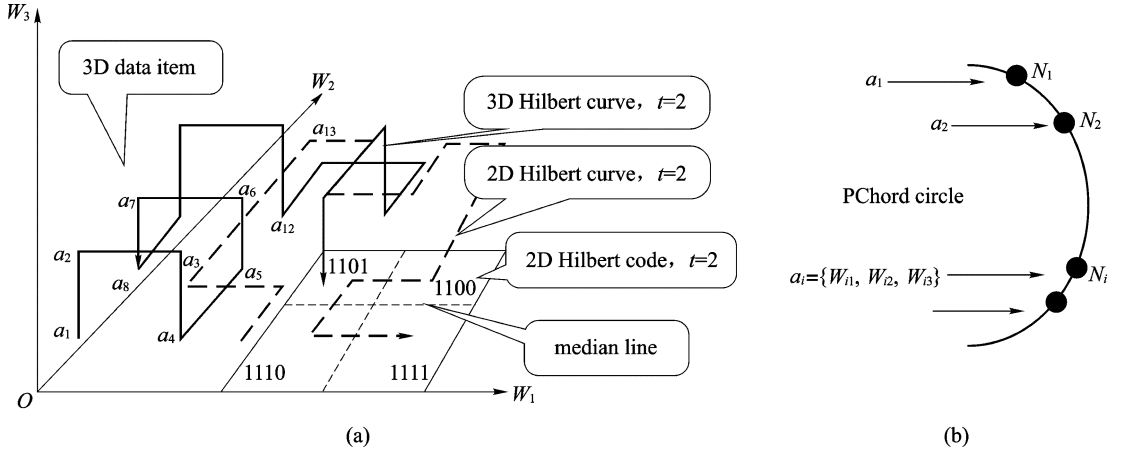

Fig. 3 3D characteristic index space and mapping relation in PChord. (a) Characteristic index space; (b) mapping relation in PChord

$$\text{Chord}(4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^{\mathrm{T}}, \tag{5}$$

$$\text{PChord}(4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \end{bmatrix}^{\mathrm{T}}. \tag{6}$$

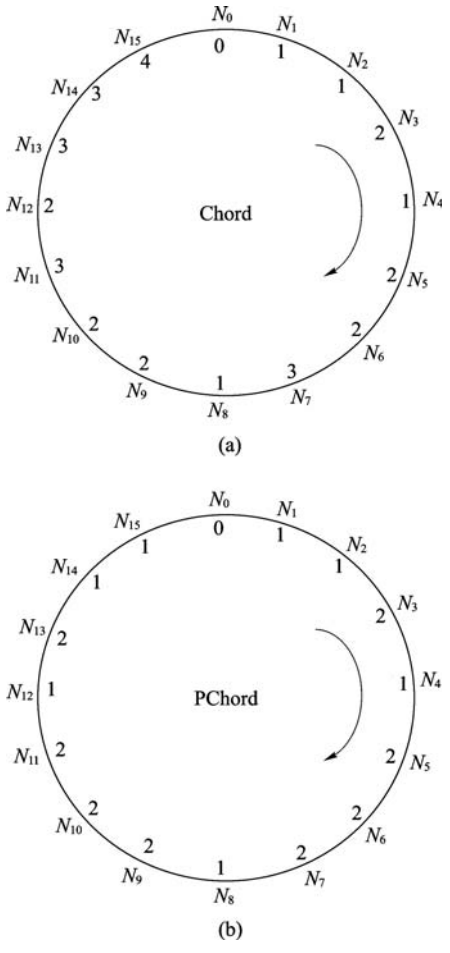

Fig. 4 Routing length comparison of Chord and PChord ($m = 4$). (a) Routing length of Chord; (b) routing length of PChord

For example, N_0 routing to N_{13}, in Chord we express as

$$13 = [0 + (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)] \mathrm{mod} 2^4,$$

and the routing length is the total number of "1", so the hops from N_{13} to N_0 are 3. While in PChord we express that as

$$13 = [0 + (0 \times 2^3 - 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0)] \mathrm{mod} 2^4,$$

and the routing length is the total number of "1" and "-1", so the hops are 2.

Assume that in Chord and PChord, the probability of any node routing to another is the same in hash space. In fact, nodes on the DHT circle are very sparse. Therefore, in order to simplify the proof, we assume that nodes in hash space are fully uniformly distributed. Before the demonstration, several variables definition should be put forward. $C(m)$ denotes the total searching length of Chord in hash space which contains 2^m nodes and $B(m)$ denotes that of PChord. $CQ(m)$ denotes the average searching length of Chord in hash space which contains 2^m nodes and $BQ(m)$ denotes that of PChord. $QR(m)$ equals the value of $BQ(m)$ divided by $CQ(m)$. $h(m)$ denotes the sum of the absolute value of all elements in the first column in PChord (see Eq. (6)).

First, we give proof on the average searching length in Chord. The hash space of $C(m+1)$ is twice of $C(m)$, and half of nodes' searching length are one hop more than the other in the 2^{m+1} hash space, so we could deduce

$$\begin{aligned} C(m+1) &= 2C(m) + 2^m \\ &= 2[2C(m-1) + 2^{m-1}] + 2^m \\ &= 4C(m-1) + 2 \times 2^m \\ &= 2^{m+1}C(0) + (m+1)2^m \\ &= (m+1)2^m. \end{aligned} \tag{7}$$

From Eq. (7), we know the average searching length in Chord is

$$CQ(m) = \frac{C(m)}{2^m} = \frac{m \times 2^{m-1}}{2^m} = \frac{m}{2}. \tag{8}$$

Next, proof on the average searching length in PChord is described as follows. As shown in the matrix of Eqs. (5) and (6), one of the primary features is that PChord can optimize the routing of Chord. For example, in Chord the

routing from N_0 to N_{13} denoted by tuple (1 1 0 1) could be optimized as (0 −1 0 1) in PChord. We would give the following optimization rules.

1) In Chord, if the highest bits are continuously "1", that is, the routing tuple (1 1 * 0 * 1 *) could be optimized as (0 * 1 0 * 1 *) in PChord. The symbol "*" means to repeat once, twice or more times.

2) In Chord, if the middle bits are continuously "1", that is, the routing (0 * 1 1 * 0 * 1 *) could be optimized as (1 0 * 1 0 * 1 *) in PChord.

3) Rules 1) and 2) should be resolved from the right end to the left end of the routing tuple, for example, tuple (1 1 1 0 1 1) would be executed reversely as (1 1 0 1 1 1).

The Chord routing information in the hash space which contains 2^4 nodes can be optimized to PChord routing information through the three rules.

After routing optimization, it is impossible that double or more "1" or "−1" will appear among adjacent nodes, that is, there exists at least one "0" between any "1" and "1", or "1" and "−1", or "−1" and "−1". Therefore, the following equation is available in terms of the first column of Eq. (6):

$$h(m) = 2^{m-1} - h(m-1). \tag{9}$$

When there are 2^m nodes in the hash space, the PChord routing optimization could be separated into three instances (see Eq. (6)).

1) If the routing tuples of the first two columns are (0 # ⋯), in which symbol "#" means "0" or "1", it is not necessary to optimize. There are 2^{m-1} instances like this.

2) If the routing tuples of the first two columns are (1 1 ⋯), it could be optimized as (0 −1 ⋯). As in this instance, the total number of "1" or "−1" in the second column is $h(m-1)$. There are $h(m-1)$ instances like this.

3) If the routing tuples of the first two columns are (1 0 ⋯), it is not necessary to optimize. There are $h(m)$ instances ($h(m) = 2^m - 2^{m-1} - h(m-1) = 2^{m-1} - h(m-1)$) like this.

The hash space of $B(m+1)$ is twice of $B(m)$, and half of nodes' searching length are one hop more than the other half, so it could be derived as follows:

$$\begin{aligned} B(m+1) &= 2B(m) + h(m+1) \\ &= 2B(m) + (2^m - h(m)) \\ &= 2[2B(m-1) + (2^{m-1} - h(m-1))] \\ &\quad + (2^m - h(m)) \\ &= 4B(m-1) + 2 \times 2^m - [h(m) + 2h(m-1)] \\ &= 2^{m+1}B(0) + (m+1)2^m \\ &\quad - [h(m) + 2h(m-1) + 4h(m-2) \\ &\quad + \cdots + 2^m h(0)] \\ &= C(m+1) - B(m). \end{aligned} \tag{10}$$

From Eq. (7) we could get

$$\begin{aligned} h(m+1) &= 2^m - h(m) \\ &= 2^m - [2^{m-1} - h(m-1)] \\ &= 2^{m-1} + h(m-1) \\ &= 2^{m-1} + 2^{m-3} + \cdots + h(0), \end{aligned} \tag{11}$$

$$\frac{h(m+1)}{h(m)} \approx 2. \tag{12}$$

Put Eq. (12) into Eq. (9) and get

$$h(m+1) \approx \frac{2}{3} \times 2^m. \tag{13}$$

The further relationship between $B(m+1)$ and $B(m)$ is given as follows:

$$\begin{aligned} B(m+1) &= 2B(m) + h(m+1) \\ &= 2B(m) + [2^m - h(m)] \\ &\approx 2B(m) + \frac{2}{3} \times 2^m. \end{aligned} \tag{14}$$

The value of $B(m)$ could be expressed by

$$\begin{aligned} B(m) &= C(m+1) - B(m+1) \\ &\approx C(m+1) - \left[2B(m) + \frac{2}{3} \times 2^m\right] \\ &\approx \frac{(m+1)2^m - \frac{2}{3} \times 2^m}{3}. \end{aligned} \tag{15}$$

Therefore, the average searching length of PChord which has 2^m nodes in the hash space is

$$BQ(m) = \frac{B(m)}{2^m} \approx \frac{m}{3} + \frac{1}{9}, \tag{16}$$

and the ratio of average searching length between PChord and Chord is

$$QR(m) = \frac{BQ(m)}{CQ(m)} \approx \frac{\frac{m}{3} + \frac{1}{9}}{\frac{m}{2}} \approx \frac{2}{3}. \tag{17}$$

4.2 Analysis on message transmission performance

Once any node joins or leaves the system, PChord needs to update the routing information and $O(\log^2 N)$ messages are transmitted each time. Assume that in Chord and PChord, any node routing to another shares the same probability in the hash space. Sometimes, in order to

maintain the consistency of the whole DHT space, it is necessary to send messages to all nodes in DHT. We may see in Figs. 5 and 6 that the messages in PChord spread faster than in Chord. The message transmission range of each node could be described as follows: Given 2^m nodes in the hash space, N_a denotes the message resource node and N_b is the message receiving node. In Chord, if $(N_b - N_a)\text{mod}2^k = 0, 1 \leqslant k \leqslant m$, and k should be large enough, the transmission range of N_b is $(N_b, N_b + 2^k)$. In PChord, if $(N_b - N_a)\text{mod}2^k = 0,\ 1 \leqslant k \leqslant m$, and k should be large enough, the transmission range of N_b is $(N_b - 2^{k-1}, N_b + 2^{k-1})$.

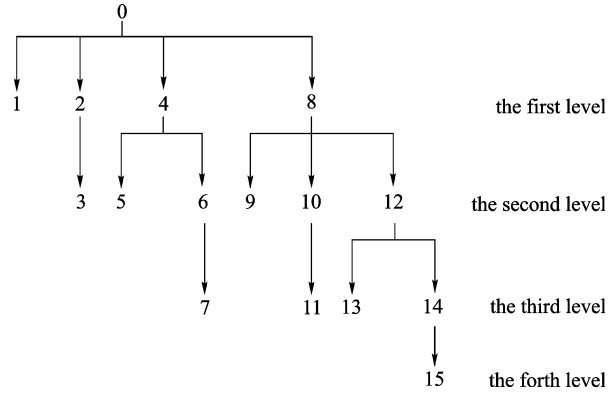

Fig. 5 Entire circle message transmission of Chord ($m = 4$)

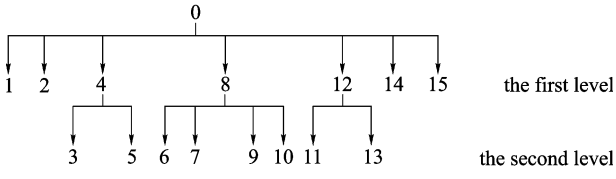

Fig. 6 Entire circle message transmission of PChord ($m = 4$)

Every time a message transmits one hop, Chord protocol can only narrow down the spread limit to half of the whole hash space. However, because of bi-directional searching, spread range would become a quarter of all nodes in PChord through one jump. To simplify the proof, assume that nodes in the hash space are uniformly distributed. Before the demonstration, it gives several variables definition. $CM(m)$ denotes the average message length of Chord in hash space which contains 2^m nodes and $BM(m)$ denotes that of PChord. $MR(m)$ denotes the ratio between the average PChord message length and the average Chord message length.

The average message length in hash space which contains 2^m nodes is essentially the longest searching length. Therefore,

$$CM(m) = m, \tag{18}$$

$$BM(m) = \frac{m}{2}. \tag{19}$$

From Eqs. (18) and (19), we could obtain that

$$MR(m) = \frac{BM(m)}{CM(m)} \approx \frac{1}{2}. \tag{20}$$

4.3 Performance analysis among seven DHT protocols

Finally, we list the performance comparison among PChord, Chord, CAN, Pastry, Tapestry, Kademlia, and Kelips protocols aimed at several types of parameters [6,11], as shown in Table 1.

From Table 1, we discover that the PChord protocol outperforms the other protocols not only on routing length and average searching length, but also on scalability and load balance.

5 Simulation results

In order to further verify the superiority of searching efficiency, message transmission and load balance ability of PChord, we made abundant experimental testing. Some contemporary P2P network simulators have been applied, such as NeuroGrid, PSim, 3LS, FreePastry, P2PSim, and so on [17]. P2PSim is an excellent simulator, which nearly supports all kinds of coverage network protocols or their extension versions, and is facilitated to verify the performance of P2P networks. P2PSim sorts events according to the sequence in which discrete events take place, and simulates the running characteristic of the actual system according to the transpirable event's impact on the system.

Table 1 Performance comparison of seven DHT protocols

parameters	PChord	Chord	CAN	Pastry	Tapestry	Kademlia	Kelips
scalability	best	best	medium	best	best	medium	medium
load balance	best	medium	best	–	worse	medium	medium
routing length	$O(\log N^{1/2})$	$O(\log N)$	$O(d)$	$O(\log_b N)$	$O(\log_b N)$	$O(\log N)$	$O(N^{1/2})$
average searching length	$O(\log N^{2/3})$	$O(\log N)$	$O(dN^{-1/d})$	$O(\log_b N)$	$O(\log_b N)$	$O(\log N)$	$O(1)$

Notes: parameter N depicts the total number of nodes, d represents one fixed value in CAN, and b denotes entries number of each node routing table for Pastry or Tapestry.

5.1 Experiments for average searching hops

In simulation experiments for average searching hops, we use different scale of nodes, the value N (nodes sum total) varies from 100 to 1000, and the number of keywords kept in each node is $100N$, $N = 100,200,...,1000$. The node identifier would be mapped to a 2^{160} hash space according to SHA.

Aimed at 10 different instances, 10000 times routing searches are carried out, and the identifier is picked stochastically as the searching start or the destination. Figure 7 compares an average searching hop of 10000 times experiments on the DHT circle from a random start to a random end using Chord, Kademlia, and PChord, separately. The simulation results prove that PChord is more efficient than Chord and Kademlia, consistent with the theoretical result in Sect. 4.1. Figure 8 shows the 10000 times running average transmission hops on the DHT circle from a random start to a random end using Chord, Kademlia, and PChord, respectively. The average message transmission hops in Chord or Kademlia is nearly twice that in PChord.

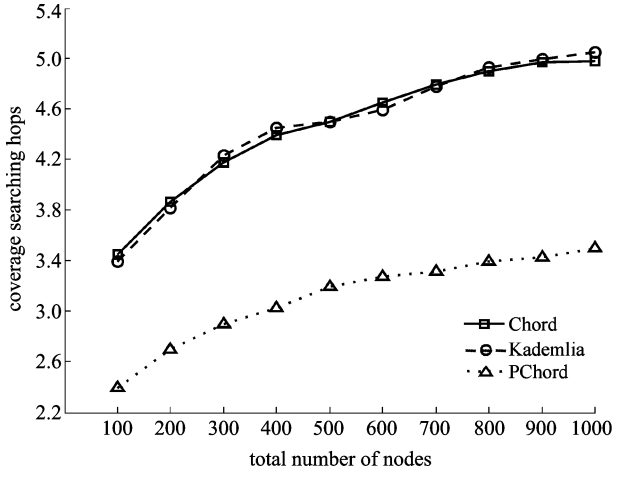

Fig. 7 Average searching hops comparison of Chord, Kademlia and PChord

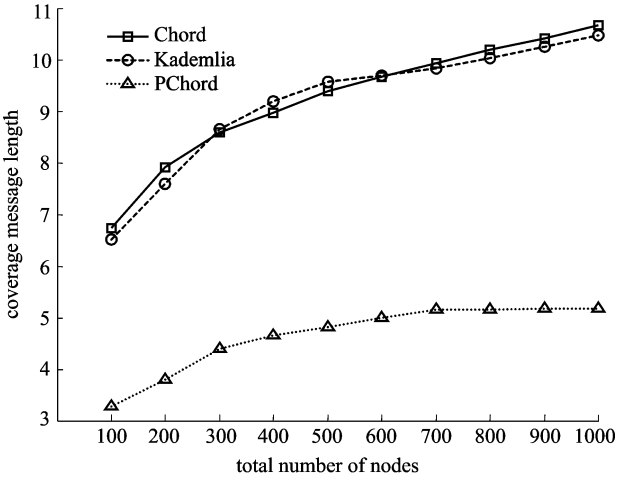

Fig. 8 Average message transmission hops comparison of Chord, Kademlia and PChord

5.2 Experiments for load balance

Experiments are considered for a network incorporating 1000 nodes, which varies the total number of keywords from 10^4 to 10^5. For each value, each corresponding simulation is repeated 10 times. Figure 9 plots the mean value of keywords per node according to different scale scenarios. For example, we could get the mean value, 67, while total number of keywords is 50000 and the satisfactory expected value is 50. Figure 10 offers stronger evidence than Fig. 9, both of whose endeavor is implemented to represent the excellent character on load balance of PChord protocol. Figure 10 focuses on how many nodes obtain the same keywords aimed at one fixed value labeled on the abscissa, in which 80000 keywords are contained in the whole network with 1000 nodes.

Figure 10 illustrates that the number of keywords of at least 600 nodes vary from 80 to 140, which proves the performance of load balance in PChord.

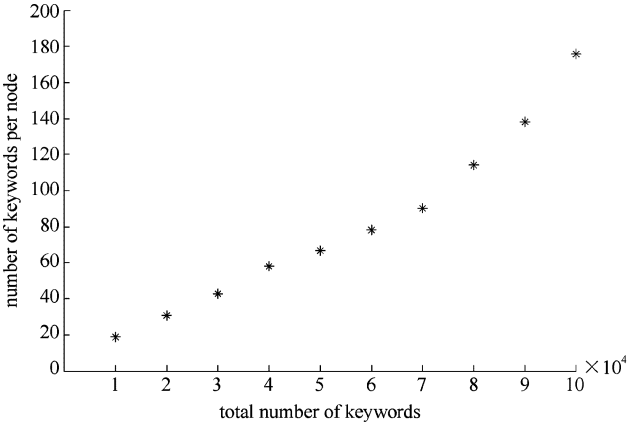

Fig. 9 Mean value of keywords per node

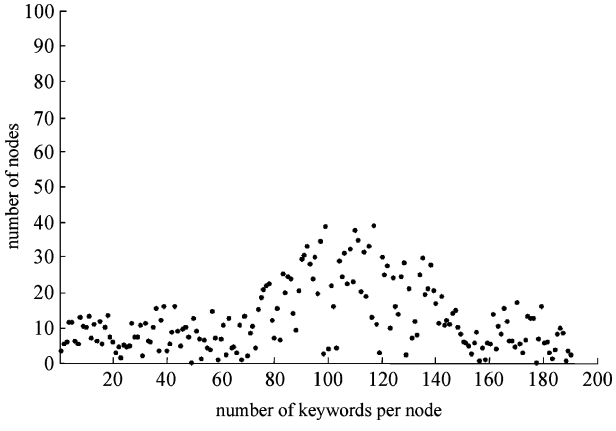

Fig. 10 Distribution of number of keywords per node

5.3 Experiments for searching validity between Chord and PChord

Some experiment parameters would be assumed, such as attribute dimensions, hierarchy value, and the total number of P2P nodes. Similar to Sect. 5.1, we use different scales of nodes, the value N varies from 100 to 1000, and the array of keywords could be represented by $a_i = \{W_{i1}, W_{i2}, W_{i3}\}$, where $d = 3$ and $t = 2$ in the PChord protocol. Figure 11 shows the average results about searching success times between two protocols at 100 times tests total for each scenario where N denotes horizontal coordinates. In each experiment, 2 ‰ identifiers would be stochastically picked where some keywords could be completely accurately matched, others be partly matched, while others be hardly matched.

It is shown in Fig. 11 that the average searching success times in PChord is more than that in Chord, because of the mechanism of approximate searching in PChord which makes some keywords be roughly matched.

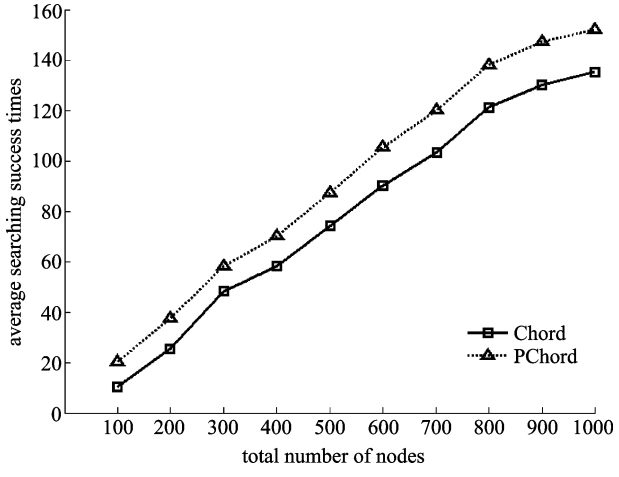

Fig. 11 Average searching success times comparison of Chord and PChord

6 Conclusions

Resource searching efficiency in structured P2P networks is one of the crucial problems to be resolved in many applications based on DHT. Based on the Chord protocol, PChord is proposed. Using bidirectional searching mechanism, the average searching length in PChord is 2/3 of that in Chord, and the average message length is about 1/2 of that in Chord or Kademlia. PChord improves the structure of the node's finger table in order to enhance internal lookup efficiency and based on Hilbert space filling curve, it realizes the multi-keyword approximate searching mechanism available for more P2P applications. Lots of simulation experiments verify the aforementioned advantages and reveal the obvious feature of load balance in PChord.

In future works, how to realize both the precise and approximate asymmetrical efficient searching in P2P networks is our target.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 60773041), the Natural Science Foundation of Jiangsu Province, China (No. BK2008451), the National High Technology Research and Development Program of China (No. 2006AA01Z219), the High Technology Research and Development Program of Nanjing City (Nos. 2007RZ106, 2007RZ127), Foundation of National Laboratory for Modern Communications (No. 9140C1105040805), Project supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 07KJB520083), Special Fund for Software Technology of Jiangsu Province, Post-doctoral Foundation of Jiangsu Province (No. 0801019C), Science & Technology Innovation Fund for Higher Education Institutions of Jiangsu Province (Nos. CX08B-085Z, CX08B-086Z), and the six kinds of Top Talent of Jiangsu Province.

References

1. Talia D, Trunfio P. Toward a synergy between P2P and grids. IEEE Internet Computing, 2003, 7(4): 94–96
2. Dumitriu D, Knightly E W, Kuzmanovic A, Stoica I, Zwaenepoel W. Denial-of-service resilience in peer-to-peer file sharing systems. ACM SIGMETRICS Performance Evaluation Review, 2005, 33(1): 38–49
3. Jia D, Yee W G, Frieder O. Spam characterization and detection in peer-to-peer file-sharing systems. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management. Napa Valley: ACM Press, 2008, 329–338
4. Zhang S F, Wu G X. Study of asymmetric DHT method and load balancing in P2P network. Journal on Communications, 2007, 28 (9): 60–67 (in Chinese)
5. Zheng Y, Chen H W, Wang R C, Wang Y. Novel distributed Hash table-BChord. Journal on Communications, 2008, 29(2): 22–28 (in Chinese)
6. Stoica I, Morris R, Karger D, Kaashoek M F, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego: ACM Press, 2001, 149–160
7. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego: ACM Press, 2001, 161–172
8. Rowstron A, Druschel P. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Berlin: Springer, 2001, 329–350
9. Zhao B Y, Huang L, Stribling J, Rhea S C, Joseph A D, Kubiatowicz J D. Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications, 2004, 22 (1): 41–53
10. Lin G C. Improvement of Kademlia-based P2P network resource locating model. Computer Engineering, 2008, 34(18): 111–112, 116

(in Chinese)

11. Chan H N, Van K N, Hoang G N. Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network. In: Proceedings of the 2nd International Conference on Communications and Electronics. Hoian: IEEE Communication Society, 2008, 126–131
12. Wu Y C, Liu C M, Wang J H. Enhancing the performance of locating data in Chord-based P2P systems. In: Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems. Victoria: IEEE Computer Society, 2008, 841–846
13. Ratnasamy S, Stoica I, Shenker S. Routing algorithms for DHTs: some open questions. In: Goos G, Hartmanis J, van Leeuwen J, eds. Proceedings of the 1st International Workshop on Peer-to-Peer Systems. Lecture Notes in Computer Science, Vol. 2429. Berlin: Springer, 2002, 45–52
14. Lewin D M. Consistent hashing and random trees: algorithms for caching in distributed networks. Dissertation for the Master's Degree. Cambridge, MA: Massachusetts Institute of Technology, 1998
15. Shao Y C, Shen D R. HilbertChord: a kind of method for managing service resources in P2P. In: Proceedings of 2008 International Conference on Computer Science and Software Engineering. Wuhan: IEEE Computer Society, 2008, 106–109
16. Xu H B, Hao Z X. K-closest pairs query algorithm based on Hilbert curve. Computer Engineering, 2008, 34(2): 17–19 (in Chinese)
17. Roverso R, Al-Aggan M, Naiem A, Dahlstrom A, El-Ansary S, El-Beltagy M, Haridi S. MyP2PWorld: highly reproducible application-level emulation of P2P systems. In: Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops. Venice: IEEE Computer Society, 2008, 272–277