2-Chord Halved

Gennaro Cordasco and Alessandra Sala
IsisLab–Dipartimento di Informatica ed Applicazioni
Università di Salerno, 84081 Baronissi (Salerno), Italy
E-mail: {sala,cordasco}@dia.unisa.it

***Abstract*— We present *2-Chord Halved*, a distributed peer-to-peer lookup protocol. This protocol, as Chord [24], uses consistent hashing [5], [7] to assign keys to nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and requires relatively little movement of keys when nodes join and leave the system.**
Our proposal exhibit the following advantages:
i) We show a *stabilization* procedure that eliminates the *fix.finger* procedure of Chord protocol. Our strategy allows to inform each node on the ring that is interested to a topological change. *Fix.finger* in Chord costs $O(\log^2 N)$ messages when it is ran on all finger table entries even if the finger table is up to date, contrariwise our stabilization procedure, that has the same cost, is ran only if there are join or leave operations and only on the interested nodes.
ii) *2-Chord Halved* is a dynamic protocol that deals with nodes joining the system and with nodes that fail or leave voluntarily. We present a new strategy to implement the join/leave operations using the predecessor's finger table of joined node and exploiting the fingers of predecessor as start point searching new fingers. This procedure costs $O(\log N \log\log N)$ w.h.p., contrariwise to Chord within join/leave operation cost $O(\log^2 N)$.
iii) We show a new routing strategy that has a moderate improvement on average path length.
The improvements are obtained with no harm to the operational efficiency (e.g. stability, scalability, fault–tolerance, node congestion) of the Chord systems.

I. Introduction

Peer-to-peer (P2P) networks is a class of networks in which each workstation has equivalent capabilities and responsibility and communications are potentially symmetric. A review of the features of the recent peer-to-peer applications yield a long list: redundant storage, permanence, selection of nearby servers, anonymity, search, authentication and hierarchical naming.

Despite of this rich set of features, the core operation in most peer-to-peer systems is efficient location of data items. Many are the recent P2P applications that are available. Among the most popular, without any doubt, are the file sharing systems and, in general, sharing other resources such as Napster [8], Gnutella [9], Kazaa and Freenet [10].

Scalability has been recognized as the central challenge in designing such systems. To obtain a scalable system, several P2P systems are based on distributed hashing table (DHT) schemes [1], [12], [14], [24], [25]. In Distributed Hash Table (DHT) schemes, data as well as nodes are associated with a key and each nodes in the system is responsible for storing a certain range of keys. Each node stores data that correspond to a certain portion of the key space, and uses a routing scheme to forward the request for data whose key does not belong to its key space to the appropriate next-hop node. The mapping table is not stored explicitly anywhere. Instead, hosts configure themselves into a structured network such that mapping table lookups require a small number of hops.

Designing a practical scheme along these lines is challenging because of the following desiderata:

Scalability: The protocol should work for a range of networks of arbitrary size.

Stability: The protocol should work for hosts with arbitrary arrival and departure times, typically with small lifetimes.

Performance: The protocol should provide low latency for hash lookups and low maintenance cost in the presence of frequent joins and leaves.

Flexibility: The protocol should impose few restrictions on the remainder of the system. It should allow for smooth tradeoffs between performance and state management complexity.

Simplicity: The protocol should be easy to understand, code, debug and deploy.

In the following we will refer to all systems that are defined on Chord as *Chord-like* systems, i.e. systems that work on ring, use a consistent hash and perform join/leave operations like Chord. In this category we can find Kademlia [3], F-Chord [4], Koorde [11], Symphony [19], randomized-Chord [21], [22], Viceroy [12].

We propose a Chord-like protocol called *2-Chord Halved*. This protocol proposes a new stabilization procedure that does not need any (costly) periodic update (such as Chord *fix.finger*), but, with a modified routing strategy, keeps Chord performance on max path length and allows a moderate improvement on average path length.

Moreover, we present a new join procedure on *2-Chord Halved* that, w.h.p.[1], takes $O(\log N \log\log N)$ messages (with respect to $O(\log^2 N)$ in Chord). *2-Chord Halved* allows to preserve consistent information between fingers of nodes when there are topological changes on the ring.

A. Related work

The measures to optimize to obtain better performance for DHTs include:

Degree: the number of neighbors with which a node must maintain continuous contact;

Hop count: the number of hops needed to get a message from any source to any destination;

[1]With High Probability, namely with probability smaller than $\frac{1}{N}$, where N is the number of nodes in the network.

The degree of fault tolerance: what fraction of the nodes can fail without eliminating data or preventing successful routing;

The maintenance overhead: how often messages are passed between nodes and neighbors to maintain coherence as nodes join and depart;

The degree of load balance: how evenly keys are distributed among the nodes, and how much load each node experiences as an intermediate node for other routes.

There are other measures for DHTs, such as delay (i.e., proximity routing) and resilience against malicious nodes.

The Chord system was introduced in [24] to allow efficient lookup in a Distributed Hash Table. Chord provides fast distributed computation of a hash function mapping keys to nodes responsible for them. Chord assigns keys to nodes with *consistent hashing* [5], [7], which has several desirable proprieties. The consistent hash function assigns each node and key an m-bit *identifier* using "SHA-1" [6] as a base hash function. Consistent hashing assigns keys to nodes as follows. Identifiers are ordered on an identifier ring modulo 2^m, labeled from 0 to $2^m - 1$. Key k is assigned to first node, called $successor(k)$, whose identifier is equal to or follows k in the identifier space (i.e. the first node clockwise from k).

To accelerate lookups, Chord maintains logarithmic routing information. Each node x maintains a routing table with up to m entries called the *finger table*. The i^{th} entry[2] in the table at node x contains the identity of the first node s that succeeds x by at least 2^{i-1} on the identifier circle, where $1 \leq i \leq m$. We denote by *n.finger[i]* the i^{th} finger of node x. The degree and the diameter are m, the average path length is $(\frac{m}{2})$. Routing is greedy, never overshooting the destination. When the number of nodes on the ring are N with $N << 2^m$ then by using logarithmic size routing tables in each node, Chord allows to find in $O(\log N)$ routing hops the node of a P2P system that is responsible for a given key.

Efficient routing in Chord is easy due to the fact that Chord is *uniform*: x is connected to $y \Leftrightarrow x + z$ is connected $y + z$. Uniformity is a crucial requirement, since it makes any system a good candidate for real implementations: besides simplicity in the implementation it also offers an optimal greedy routing algorithm without node congestion [26].

Adding or removing a node is accomplished at a cost of $O(\log^2 N)$ messages, in fact each node periodically calls *fix.fingers* to make sure its finger table entries are correct; this is how new nodes initialize their finger tables, and it is how existing nodes incorporate new nodes into their finger tables. By way of explanation each *fix.finger* cost $O(\log N)$ messages and we need $\log N$ round of *fix.finger* to initialize all the finger table.

Tapestry [25] adapted this scheme to a dynamic network for use in a global data storage system. Pastry [1] is another scheme along the same lines where a node forwards a query to a neighbor with the longest matching prefix. In both Tapestry and Pastry, the number of bits per digit b is a configurable parameter that remains fixed at run-time.

[2]All the arithmetic operation on the Chord's ring are done $\mod 2^m$.

CAN [14] embeds the key-space into a torus with d dimensions by splitting the key into d variable-length digits. A node forwards a query to the neighbor that takes it closer to the key. Nodes have $O(d)$ neighbors and routing latency is $O(dN^{1/N})$. The number of dimensions d is fixed in CAN. If the final network size can be estimated, then d could be made $O(\log N)$, resulting in $O(\log N)$ routing latency and $O(\log N)$ neighbors.

Viceroy [12] is the first proposal that provides $O(\log N)$ routing latency with only a constant number of links. Like Chord, nodes are placed along a circle. A node additionally belongs to one out of approximately $O(\log N)$ concentric rings lying one above the other. These rings correspond to layers in Butterfly networks. A node maintains connections with two neighbors each along the two rings it belongs to. It also maintains two connections to a pair of nodes in a lower ring and one connection with a node in the ring above. Routing requires $O(\log N)$ hops on average.

Another important step toward protocols P2P with higher performance came from Milgram's idea. The *small world phenomenon* was discovered by Milgram [16] in a celebrated experiment that demonstrated that pairs of people in a society were connected by short chains of acquaintances. Milgram also discovered that people were actually able to route letters to unknown persons in a few hops by forwarding them through acquaintances. To model the small world phenomenon, Kleinberg [17] recently constructed a two dimensional grid where every node maintains four links to each of its closest neighbors and one *long-distance* link to a node chosen from a harmonic probability distribution. In the resulting network, a message can be routed from any node to another by greedy routing in $O(\log^2 N)$ hops on average. Barriere et al. [18] studied Kleinberg's construction and proved its optimality under certain conditions. In [19] Manku et al. extend Kleinberg's result by showing that with $k = O(1)$ links, the routing latency diminishes to $O(1/k \log^2 N)$ hops. They also shows how this basic idea can be adapted and engineered into a practical protocol for maintaining DHTs in a peer to peer network.

Others randomized P2P networks include randomized-hypercubes [20], [21], randomized-Chord [21], [22], and a combination of Kleinbergs construction with butterfly networks [23].

B. Our Result

We introduce here a Chord-like protocol called *2-Chord Halved*. In Section III we proposes a new stabilization procedure, that does not need any (expensive) periodic update (such as Chord *fix.finger*), but, with a modified routing strategy that is shown in Section IV, keeps Chord performance on max path length and allows a moderate improvement on average path length.

Moreover, in Section V, we present a new join procedure on *2-Chord Halved*, that takes $O(\log N \log\log N)$ messages w.h.p. (with respect to $O(\log^2 N)$ in Chord). Section VI concludes the paper with some final remarks.

2-Chord Halved allows to preserve consistent information between fingers of nodes when there are topological changes on the ring. The new stabilization procedure acquaints each node on the ring that is interested to a topological change. Stabilization is ran, in *2-Chord Halved*, only when there is a join or leave operation and only on interested nodes, contrariwise to Chord that needs periodic update on all nodes. The idea is to use the fingers of predecessor's node that has joined or has leaved the network to stabilize the network. This strategy eliminates Chord's *fix.finger* that performs the same utility but with higher cost. Indeed, the *fix.finger* procedure is ran on periodic interval time and for each time, on a certain amount of finger table entries. The *fix.finger* procedure costs $O(\log^2 N)$ when it is ran on all finger table entries even if the finger table is up to date, contrariwise our stabilization procedure, that have the same cost, is ran only if there are join or leave operations and only on the interested node.

2-Chord Halved defines a new finger table where the fingers are nodes that have a distance equal to even powers of two, so, the routing algorithm can profit by edges in both directions. In particular, when m is odd, we can exhibit a moderate improvement, on the average path length.

We present a new strategy of routing which implements the idea to use both clockwise fingers and anticlockwise fingers in the routing path. We use a tool provided by Manku et al. [2] that solves our routing problem (i.e. using edges in both directions on ring but having only fingers with even power of two), without paying overhead, i.e. the longest path has length m and the average path length is less than[3] or equal to $\frac{m}{2}$.

The last part of this paper deals with the join operation and how it can be improved. When a node wants to be connected to the system, it computes its "finger table", and, only then, it can take part to the routing algorithm. In Chord [24], when a node want to be connected to the system, it must initialize its finger tables. The cost of the initialization is $O(\log^2 N)$ because the *fix.finger* procedure is executed for all the entries of the finger table.

We present a new strategy to implement the join operation that costs $O(\log N \log\log N)$, w.h.p., using the idea to take advantage from predecessor's finger table. With high probability, the number of nodes, in the network, that must be found to construct the "finger table" is $O(\log N)$, and everyone takes $O(\log\log N)$ steps. We want to stress that the new join procedure does not add any overhead, and that the strategy is adaptable for all Chord-like uniforms protocol, such as *Chord* [24], *F-Chord* [4] and in general for all possible uniform protocols on a ring.

II. Preliminary results

2-Chord Halved holds a set N of nodes lying on a ring of 2^m identifier (labeled from 0 to 2^m-1 in clockwise order). Each node x, has an m bit ID and is connected with its predecessor $p(x)$ and its successor s(x) on the ring.

[3]In particular if m is odd the average path length of our scheme is less than $\frac{m}{2}$.

Let x and y two nodes we denote by $\delta(x,y)$ be the clockwise distance between x and y on the ring, instead, let $d(x,y)$ be the shortest path to go from x to y. Due the fact that *2-Chord Halved* has front and back fingers, we obtain a *symmetric* system, where it is easy to show that the follow is true:

$$d(x,y) = d(y,x)$$

Let *A* be the event that the distance between two generic consecutive nodes is almost $\alpha = \frac{2^m \log N}{N}$. The first theorem bounds from above the probability of the event *A* occurs.

Theorem 1: With high probability, the maximum distance between two generic consecutive nodes is $\alpha = \frac{2^m \log N}{N}$ where the number of nodes alive is $N < 2^m$, namely

$$Pr[\exists x \text{ s.t. } \delta(x, s(x)) > \alpha] < \frac{1}{N}.$$

Proof: Since x and $s(x)$ are two consecutive nodes then between x and $s(x)$ there are no other nodes lying in a range $2^m - \delta(x, s(x))$. Thus

$$\begin{aligned} & Pr[\exists x \text{ s.t. } \delta(x,s(x))>\alpha] \\ = \; & Pr[\text{there are } N \text{ nodes in } 2^m-\alpha \text{ ID}] \\ = \; & \prod_{i=0}^{N-1} \frac{2^m-\alpha-i}{2^m} < \left(\frac{N-\log N}{N}\right)^N < \frac{1}{N} \end{aligned}$$

■

Let *B* be the event that in a range α the number of nodes is at most $4\log N$. Now we want to upper bound the probability of event *B* given that event *A* is holds true.

Theorem 2: With high probability, the maximum number of nodes that lie in a range of size $\alpha = \frac{2^m \log N}{N}$, knowing that in a generic α range there are no nodes, is $4\log N$, thus $Pr(B/A) < \frac{1}{N}$ where the number of nodes alive is $N < 2^m$.

Proof: We consider a generic interval I of size α. Let X_i be an independent variable such that

$$X_i = \begin{cases} 1, \text{ if the } i^{th} \text{ node belongs to } I \\ 0, \text{ otherwise} \end{cases}$$

so let $p_i = Pr[X_i = 1] = \frac{\alpha}{2^m-\alpha} = \frac{\log N}{N-\log N}$.
Let $X = \sum_{i=1}^{N} X_i$ and $\mu = E[X] = E\left[\sum_{i=1}^{N} X_i\right] = \frac{N\log N}{N-\log N}$.
According to *Chernoff Bound* [13] we can show the follow:
Let $\rho = 4\frac{N-\log N}{N} - 1$ we have:

$$\begin{aligned} Pr[X > 4\log N] &= Pr[X > (1+\rho)\mu] \\ &< \left(\frac{e^\rho}{(1+\rho)^{(1+\rho)}}\right)^\mu \\ &= \left(\frac{e^{4\frac{N-\log N}{N}-1}}{\left(4\frac{N-\log N}{N}\right)^{\left(4\frac{N-\log N}{N}\right)}}\right)^{\frac{N\log N}{N-\log N}} \\ &< \left(\frac{e}{4\frac{N-\log N}{N}}\right)^{4\log N} < \frac{1}{N}. \end{aligned}$$

■

It is easy to show that:

Corollary 1: $Pr(A \cap B) < \frac{1}{N^2}$, namely given an empty interval of size $\alpha = \frac{2^m \log N}{N}$ for each generic interval of size α there are at most $4 \log N$ nodes w.h.p. where the number of nodes alive is $N < 2^m$.

Now we can consider that the maximum distance between two consecutive node is $\alpha = \frac{2^m \log N}{N}$ and the maximum number of nodes within that distance are $4 \log N$ w.h.p.

The next theorem shows that $\beta = \frac{2^m}{N^2}$ is the minimum distance between two generic consecutive nodes on the ring.

Theorem 3: With high probability, the minimum distance between two generic consecutive nodes is $\beta = \frac{2^m}{N^2}$ where the number of nodes alive is $N < 2^m$, namely

$$Pr[\exists x \text{ s.t. } \delta(x, s(x)) < \beta] < \frac{1}{N}.$$

Proof: If x and $s(x)$ are two consecutive nodes then between x and $s(x)$ there are no other nodes. Thus

$$\begin{aligned} Pr[\exists x \text{ s.t. } \delta(x, s(x)) < \beta] &= 1 - Pr[\forall x\ \delta(x, s(x)) \geq \beta] \\ &= 1 - \prod_{i=0}^{N-1} \frac{2^m - \beta}{2^m} \\ &< 1 - \left(1 - \frac{1}{N^2}\right)^N < \frac{1}{N} \end{aligned}$$

■

III. New stabilization procedure in *2-Chord Halved*

In this section we present the fingers of *2-Chord Halved* and the new stabilization procedure. As is usually, also in *2-Chord Halved*, the identifiers are ordered on an *identifier circle* modulo 2^m, and, the key k is assigned to the first node whose identifier is equal to or follows k in the identifier space. The fingers in *2-Chord Halved* are m like in Chord but the fingers, into new protocol, point a different nodes than Chord. In fact *2-Chord Halved* uses only fingers with even power of two index but in both directions on the ring.

Each node x maintains a routing table with up m entries as follows:

- *Front finger*

$$x.finger[i] = (x + 2^{2i}) \ \ \forall\ 0 \leq i < \left\lceil \frac{m}{2} \right\rceil$$

- *Back finger*

$$x.finger[i] = (x - 2^{2i}) \ \ \forall\ 0 \leq i < \left\lfloor \frac{m}{2} \right\rfloor$$

After we show the new *stabilization* procedure that eliminates the *fix.finger* procedure of Chord protocol. This strategy allows to inform each node on the ring that is interested to a topological change. Stabilization is always ran when there is a join or leave operation. If there is a join or leave operation our scheme is able to identify and inform groups of nodes that will have some fingers uncorrect. We propose a strategy to recognize this groups of nodes and, so, our system can change only those fingers of the nodes that need to change own values. In particular everyone of this groups of nodes have only one finger uncorrect and, so, to estimate the complexity of this procedure we need to know the number of nodes that are lying into those groups and we show that this number is $O(\log^2 N)$. We need to inform all this nodes into stabilization procedure, and so, the complexity of messages of this procedure is $O(\log^2 N)$. The proposed improvement with *2-Chord Halved* is to offer a dynamic network that fit oneself efficiently to joined or leaved nodes without allow inconsistent information of routing.

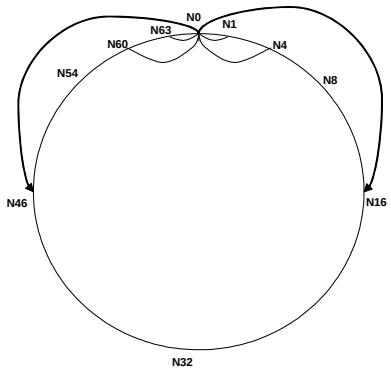

Fig. 1. *2-Chord Halved* with $m = 6$.

A. Definitions and theorems

Let x be a generic nodes on the ring, let y=s(x) and let $0 \leq j < \left\lceil \frac{m}{2} \right\rceil$, we denote by $I_j^+(x, y)$ (resp. $I_j^-(x, y)$) the interval $\left]x - 2^{2j}, y - 2^{2j}\right]$ (resp. $\left]x + 2^{2j}, y + 2^{2j}\right]$), as shown in figure 2.

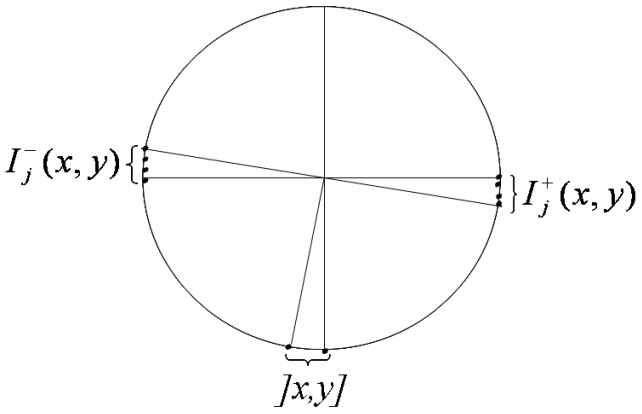

Fig. 2. In figure has been identified $I_j^+(x, y)$ and $I_j^-(x, y)$ with $j = \lfloor \frac{m}{2} \rfloor - 1, when \text{m} is even.$

It is easy to see that for each node k belonging to $I_j^+(x, y)$ (risp. $I_j^-(x, y)$) the j^{th} front (risp. back) finger of k belongs to $]x, y]$ i.e., for each node $k \in I_j^+(x, y) \Rightarrow x < k + 2^{2j} \leq y$ and for each $k \in I_j^-(x, y) \Rightarrow x < k - 2^{2j} \leq y$. We can observe also that for each j all the intervals $I_j^-(x, y)$, $I_j^+(x, y)$ and]x,y] have the same size.

In order to evaluate the cost of the stabilization procedure it is necessary to upper bound number of these interval and to know the number of nodes that lie into them. We have analyzed that the range of the considered interval don't influence asymptotically the number of groups that have fingers into that interval. We prove that even if the distance, between two consecutive nodes, is highest (i.e. α), or if the distance is smallest (i.e. β) the number of groups, that have a finger in $]x, y]$, will be $O(\log N)$ w.h.p..

Lemma 1: The number of groups are inversely proportional to distance between x and y (i.e between two consecutive nodes) that is $\beta \leq \delta(x, y) \leq \alpha$ instead the number of nodes into these groups are proportional to the greatness of these.

According to the Theorems 1 and 3, and according to the Lemma 1 we use in the first part of following theorem the minimal distance between two consecutive nodes, instead in the second part the maximal distance, so we consider always the worse case.

Theorem 4: Let x a generic node of the ring and y=s(x), then on the ring there are $O(\log N)$ groups of nodes that have a finger in $]x, y]$ and the total number of nodes in these intervals is $O(\log^2 N)$.

Proof: The proof is articulated into two part:

- We need to find the number of intervals like $I_j^+(x, y)$ and $I_j^-(x, y)$. We start by considering an interval of 2^{m-1} *ID* (half ring) and each time we encounter a new interval $I_j^+(x, y)$ we divide by 4 the remaining interval to encounter $I_{j-1}^+(x, y)$.
  Thus let k be the distance between x and y, then the ring will can be divided i times until $\frac{2^{m-1}}{4^i} \geq k$. Therefore the number of times that we trace these groups is $4^i = \frac{2^{m-1}}{k}$ thus: $i = \frac{(m-1)}{2} - \frac{\log k}{2}$. In this case we use the minimal distance to have an upper bound on the number of groups. If $k = \beta = \frac{2^m}{N^2}$ then $i \leq \log N - 1/2$.
  By the same argument we can show that the number of intervals on the latter part of the ring is at most $\log N - 1/2$, so the total number of groups are $2 \log N - 1$.
- We need to find the number of nodes that there are into previous groups. According to the Theorem 2 and Lemma 1 we say that for every interval of size α there are almost $4 \log N$ nodes and, so, the total number of nodes into all groups is at most $8 \log^2 N - 4 \log N$.

■

Now we know the number of nodes to find during stabilization procedure. Everyone of groups above identified have a only one finger in $]x, y]$ and so if between $]x, y]$ there is a topological change only that finger will be updated. Indeed, all times that there is a operation of join or leave these groups will be updated everyone changing only one particular finger for all nodes into them. Interesting is that these groups are easy to find consulting the predecessor's finger table of the joined/leaved node. The system is symmetric and so we reaches the nodes that aim to $]x, y]$ with front fingers using the predecessor's back fingers and vice versa. We remember that in Chord [24], each node periodically calls, for every node, the *fix.finger* procedure to make sure its finger table entries a correct; the cost of this procedure is not influenced by some join or leave event but is a continuous cost of Chord protocol. Our stabilization procedure, on the contrary, pays $O(\log^2 N)$ only if there is a topological change and only on the interested nodes.

IV. Routing in *2-Chord Halved*

Efficient routing in our scheme is easy due to the fact that 2-Chord Halved is uniform: x is connected to $y \Leftrightarrow x + z$ is connected to $y + z$.

In spite of the fact that our scheme is different by Chord because it holds fingers in both directions, it offers the same performance of Chord on max path length and a little bit better on average path length.

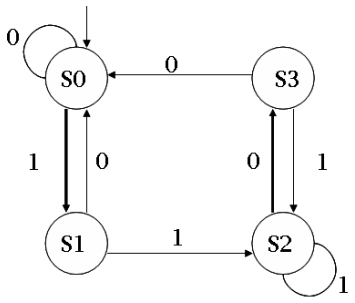

Fig. 3. The state machine used by automata based algorithm that solves the problem in definition 1, as presented in [2].

We present a new strategy of routing which implements the idea to use both front fingers and back fingers in the routing path. We use a tool provided by Manku et al. [2] that solves the following problem:

Definition 1: Given $b \geq 1$ and $0 < d < 2^b$, identify (d', d'') such that $H(d') + H(d'')$ is minimal, subject to two constraints[4]:

(i) either $d = d' - d''$ OR $2^b - d = d'' - d'$;
(ii) both $d', d'' \in [0, 2^b)$.

We use a deterministic finite automata presented in [2] for solving this problem. The routing algorithm runs the automata in figure 3 for exactly b steps, where b is the number of bit to represent the identifiers of network, and in this particular case is the binary representation of distance d between the source and the destination possibly padded with leading 0s to make it exactly b bits long.

The binary representation of distance d is fed as input right–to–left. Each transition produces a pair of bits as output, the first bit for d' and the second for d''. All thin edges produce $<0, 0>$ as output. Thick edges ($S_0 \xrightarrow{1} S_1$ and $S_2 \xrightarrow{0} S_3$) produce $<0, 1>$ or $<1, 0>$ as output [2]. Observe that the traversal of a thick edge corresponds to exactly one 1-bit in either d' or d''. We can see in [2] that if we use all the Chord's jumps in both directions the path length for a given distance

[4]Where $H(x)$ denotes the Hamming norm of x, i.e., the number of 1-bits in x.

d is just the number of thick edges traversed by the automata when processing the b-bit binary representation of d. Now we show how to run routing strategy into our protocol. We use the automata above, calculating the distance d between the source of the message and its destination, the distance d is passed to the automata, and the path length of a route produced for distance d is function of the number of thick edges traversed when scanning the binary representation of d right-to-left. Indeed, in *2-Chord Halved* we have not the matching one to one between thick edges (i.e. the bits to one into (d', d'')) and jumps, so, routing analysis for its is different from [2]. The average path length is calculated as follows:

$$APL = \frac{\sum_{x,y \in N} d(x,y)}{N^2} = \frac{\sum_{x \in N} d(0,x)}{N}$$

where the second equality is due the fact that our scheme is uniform. Then computing the average path length simply requires us to run this automata on all possible b-bit binary strings and compute the average number of thick edges traversed. We define a matrix *M* $(2^b X b)$, as shown in figure 4-c which contains for each row $0 \leq i < 2^b$ the string $d'\ OR\ d''$ obtained when the automata processed the binary representation of i, that is shown in figure 4-a.

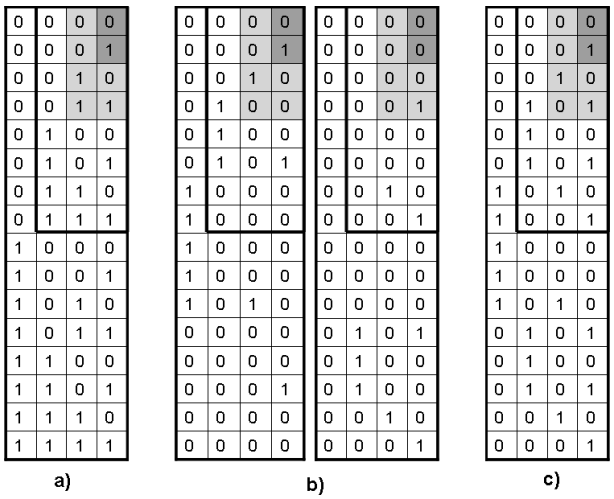

Fig. 4. a) Identifies the matrix of all permutations of $b = 4$ bit; b) the two matrix represent the output of the automata, shown in figure 3, respectively d' and d''; c) represent the matrix *M*, i.e. the OR of d' and d''. In the matrixes above has been underlined the respective submatrix with $b = 3$, $b = 2$ and $b = 1$ bit.

In theorem 5 a recurrence is shown that defines the number of one in the first column of the matrix $d'\ OR\ d''$.

Theorem 5: The number of bit to "1" in the first column of the matrix *M* is given by:

$$f(b) = 2^{b-1} - f(b-1),$$

where f(0)=0.

Proof: (Sketch) The relation is true considering that the number of one is equal to the number of zero that there are in the first column constructed with $b-1$-bit, that is exactly $2^{b-1} - f(b-1)$, because the automata produces a bit to one only if previous computational has produced a bit to zero as we can see looking at the automata's state. ■

Lemma 2: Let $f(b)$ and $f(b+1)$ be the total number of bit to "1" in the first column of the matrix *M* of all binary string that the automata computes on b-bit and $(b-1)$-bit, then:

$$f(b) + f(b+1) = 2^b$$

Proof: According to theorem 5, the proof is trivial by induction on $f(b)$. ■

We need to present another two property of the *2-Chord Halved* to analyze the average path length.

Let $E(b)$ (risp. $D(b)$) be the number of the bit "1" site in even (risp. odd), right-to-left counting from 0, column of the matrix *M*.

Lemma 3:

$$E(b) = \begin{cases} 2E(b-1), & \text{if b is odd;} \\ 2E(b-1) + f(b), & \text{if b is even} \end{cases}$$

Proof: The result, stored into matrix M, of the automata on all possible b-bit binary strings is equal to result of two times the automata on all possible $b-1$-bit, only the last, right-to-left, column is unknown; this is true because the automata is deterministic and on same prefix computes the same result. To know the total number of "1" in even positions (right-to-left) of matrix is possible to double the value obtained on $(b-1)$-bit matrix and to add $f(b)$ value only if b is even. ■

In the same mode, with only the exception that $D(b)$ counts the odd positions of bit to one, it is easy to show that

Lemma 4:

$$D(b) = \begin{cases} 2D(b-1), & \text{if b is even;} \\ 2D(b-1) + f(b), & \text{if b is odd.} \end{cases}$$

Proof: According to lemma 3 remembering that $f(b)$ is considered only if b is odd. ■

Let s and t two node at distance d and let $\hat{d} = d'ORd''$, than the node s can reach the node t by performing a jump (front or back) for each "1" in the binary representation of $\hat{d}$. In particular for each "1" in position i we need to perform the jump (front or back) 2^i. Since in our system we do not have the jumps 2^i with i odd, we simulate this jumps with two jumps of size 2^{i-1}. Hence in order to value the average path length we analyze the matrix *M*, counting the number of "1" and double counting the "1" in even columns. Indeed, the APL is valued as following:

$$APL = \frac{\sum_{x \in N} d(0,x)}{N} = \frac{D(m) + 2E(m)}{2^m}.$$

The follow theorem shows that in *2-Chord Halved* the average path length is better than Chord if m is odd.

Theorem 6: The average path length calculated with *m*-bit is:

$$\frac{D(m) + 2E(m)}{2^m} \leq \frac{m}{2}$$

Proof: By induction:

Base: $m = 1$

$D(1) = 1$, $E(1) = 0$ then $\frac{1+0}{2} = \frac{1}{2} = \frac{m}{2}$.

The inductive hypothesis is: $\frac{D(i)+2E(i)}{2^i} \leq \frac{i}{2}, \forall i < m$.
Inductive step: we want to show that:

$$\frac{D(m)+2E(m)}{2^m} \leq \frac{m}{2},$$

we have two cases:

1) if m is even :

$$\begin{aligned} & \frac{D(m)+2E(m)}{2^m} \\ = & \frac{2D(m-1)+4E(m-1)+2f(m)}{2^m} \\ = & \frac{4D(m-2)+2f(m-1)+8E(m-2)+2f(m)}{2^m} \\ = & \frac{D(m-2)+2E(m-2)}{2^{m-2}} + \frac{f(m-1)+f(m)}{2^{m-1}} \\ \leq & \frac{m-2}{2}+1 = \frac{m}{2} \end{aligned}$$

2) if m is odd:

$$\begin{aligned} & \frac{D(m)+2E(m)}{2^m} \\ = & \frac{2D(m-1)+4E(m-1)+f(m)}{2^m} \\ < & \frac{m-1}{2}+\frac{1}{2} = \frac{m}{2} \end{aligned}$$

In the second case we want to stress the improvement that the average path length of our system has with respect Chord. Indeed, when the number of bits of the system m is odd then the average path length of *2-Chord Halved* is less than $m/2$. ■

V. New join operation in *2-Chord Halved*

In order to ensure that *2-Chord Halved* is a dynamic protocol, it needs to deal with nodes joining the system and with nodes that fail or leave voluntarily. The operations that implements this functionality are join and leave. When a node wants to be connected to the system, it computes its "finger table", and, only then, it can take part to the routing algorithm. In this section we want to present a new idea to implement a join operation that holds better message complexity than Chord. In Chord [24] indeed, when a node wants to be connected to the system, it must initialize its finger tables. The cost of the initialization is $O(\log^2 N)$ w.h.p. because the *fix.finger* procedure is executed for all entries of finger table. In our scheme and in general for uniform systems, the bootstrap of a node x can easily be improved by using the finger table of x's predecessor as a starting point to build an updated finger table. In fact, the i^{th} finger of x is efficiently obtained by asking the i^{th} finger of x's predecessor. This procedure costs $O(\log N \log\log N)$ w.h.p., because with high probability, the number of nodes, in the network, that must to be found to construct the new "finger table" is $O(\log N)$, and everyone takes $O(\log\log N)$ steps w.h.p. and only $O(1)$ step on average. We want to present this new strategy stressing that the model does not add any overhead, and that this strategy is adaptable for all Chord-like uniforms protocol.

A. Idea and Formalization

Each node x, in the network, creates its own "finger table" with the support of the finger table of x's predecessor. The search of each i^{th} finger of the node x starts from the i^{th} finger of x's predecessor. This scheme bounds the range of the search, for each finger, to a distance d that is equal to the distance between x and x's predecessor. In order to ensure that the new join improves the computational time it is necessary to show the maximum distance between two consecutive nodes and the maximum number of nodes within that distance. According to theorems 1 and 2, and according to corollary 1 we can consider that w.h.p. the maximum distance between two consecutive node is $\alpha = \frac{2^m \log N}{N}$ and that the maximum number of nodes within an interval of size α is $4\log N$ w.h.p..

B. FindFinger

We are suggesting a new routine for the construction of the finger table of a node joined. The finger table contains rooms for *m* entries and for each entry we use a routine *FindFinger* that costs $O(\log\log N)$ hops. *FindFinger* executes the procedure Chord's *FindSuccessor* in a range of size at most α.

Theorem 7: Let $N > 6$, the number of hops to route a message at distance α is $O(\log\log N)$ w.h.p..

Proof: We say that after each step of the procedure *FindSuccessor* [24] the distance decrease at least with a factor of $\frac{1}{2}$. Then after $2\log\log N + 2$ exponential steps on a range at most α the remaining interval is:

$$\alpha' = \left(\frac{2^m \log N}{N} \frac{1}{2^{2\log\log N+2}}\right) = \frac{2^{m-2}}{N \log N}$$

We are investigating analyze the number of nodes that there are into α', and so, the probability that the number of nodes in $\alpha' \geq \log\log N$ conditioned that in α there are at most $4\log N$ nodes is :

$$\sum_{i=\log\log N}^{4\log N} \binom{4\log n}{i} \left(\frac{\alpha'}{\alpha}\right)^i \left(1-\frac{\alpha'}{\alpha}\right)^{4\log N - i} \overset{N\geq 6}{<} \frac{1}{N}.$$

Hence after $2\log\log N + 2$ forwarding, the distance between the current query node and the destination will be reduced to at most α', in which there are at most $\log\log N$ nodes with high probability. Thus, the next $log\log N$ forwarding steps will find the needed node. Finally in $3\log\log N + 2$ hops the destination is reached and so a *FindFinger* is ran in $O(\log\log N)$ hops. ■

This analysis allows us to say that the new join needs m execution of *FindFinger* and, so, the total cost is $O(m\log\log N)$.

C. Analysis of the new join with trivial finger

Not all the fingers of a node are different as shown in figure 5, (as an example, consider the finger table for node 8 in figure 5). For each node x, we call *trivial* fingers all the fingers that falls between x and x'successor or x's predecessor and x. Since the minimal distance between two consecutive

node is $\beta = \frac{2^m}{N^2}$ w.h.p. (see Theorem 3), we have that, the i^{th} finger (risp. back finger) of a node, for $i \leq \frac{m-2\log N}{2}$, will be equal to x' successor (risp. x) w.h.p. and then it does not need to be searched not stored separately. Thus, the trivial fingers are at least $2\frac{m-2\log N}{2} = m-2\log N$ w.h.p..

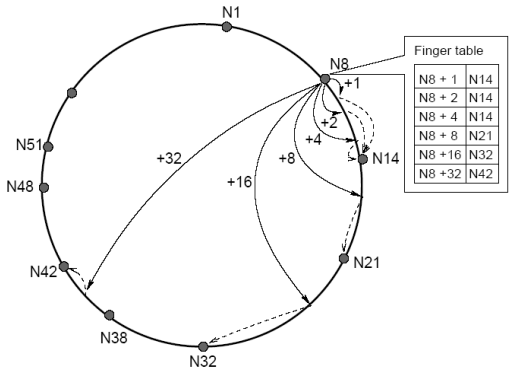

Fig. 5. The finger table entry for node N8.

Now we can recalculate the cost of the join operation considering the fact that there are at least $m - 2\log N$ trivial fingers w.h.p..
Since the non trivial fingers are $m - m - 2\log N = 2\log N$ w.h.p., for each of these we pay $O(\log\log N)$ messages to find its into *FindFinger* procedure, then to construct the finger table of a new node we need, in total, $O(\log N \log\log N)$ messages, w.h.p..

VI. Conclusions

2-Chord Halved protocol ranks into environment of the distributed peer-to-peer protocol that offers a powerful primitives (as Chord [24]): give a key, it determines the node responsible for storing the keys value, and does so efficiently. *2-Chord Halved* uses consistent hashing [5], [7] to assign keys to nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and requires relatively moderate movement of keys when nodes join and leave the system. *2-Chord Halved* evolves the idea to stabilize the networks, i.e. allows to preserve consistent information between fingers of nodes when there are topological changes on the ring. This procedure not need any (costly) periodic update (such as Chord *fix.finger*), and it is ran alone when there are topological changes. Moreover, we have presented a new join procedure on *2-Chord Halved* that, w.h.p., takes $O(\log N \log\log N)$ messages (with respect to $O(\log^2 N)$ in Chord) and a new routing strategy that keeps Chord performance on max path length and is moderately better than Chord on average path length.

Acknowledgements. The authors gratefully acknowledge Alberto Negro and Vittorio Scarano for many helpful comments and constructive suggestions.

References

[1] P. Druschel and A. Rowstron, "*Pastry: Scalable, distribute object location and routing for large-scale peer-to-peer systems*". Proc. of 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01), Nov 2001.
[2] P. Ganesan and G. S. Manku, "*Optimal Routing in Chord*". Proc. of 15th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA), Jan 2004.
[3] Petar Maymounkov and David Mazires, "*Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*". Proc. of IPTPS, Cambridge, MA, USA, Mar 2002
[4] G. Cordasco, L.Gargano, M.Hammar, A.Negro and V.Scarano, "*F-Chord: Improved Uniform Routing on Chord*". In Proc. of 11th Colloquium on Structural Information and Communication Complexity (Sirocco), Jun 2004.
[5] D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin and R. Panigrahy, "*Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on World Wide Web*". In Proc. of the 29th Annual ACM Symposium on Theory of Computing, May 1997.
[6] FIPS 180-1. *Secure Hash Standard*. U.S. Department of Commerce/ NIST, National Technical Information Service, Springfield, VA, Apr 1995
[7] D. Lewin, "*Consistent hashing and random trees: Algorithms for caching in distributed networks*". Masters thesis, Department of EECS, MIT, 1998. Available at the MIT Library, http://thesis.mit.edu/.
[8] Napster. "*http://www.napster.com/*".
[9] Gnutella. "*http://www.gnutella.com/*".
[10] I. Clarke, T. Hong, S. Miller, O. Sandberg and B. Wiley. "*Protecting Free Expression Online with Feenet* ". IEEE Internet Computing, 2002.
[11] M.F. Kaashoek and D.R. Krager, "*Koorde: A simple degree-optimal distributed hash table*". Proc. of the 2st International Workshop on Peer-to-Peer Systems (IPTPS), Feb 2003.
[12] D. Malkhi, M. Naor, D. Ratajczak, "*Viceroy: A Scalable and Dynamic Emulation of the Butterfly*". Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC), Aug 2002.
[13] R. Motwani, P. Raghavan, "*Randomized Algorithms*". Cambridge University Press, 1995.
[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "*A scalable content-addressable network*". Proc. of ACM SIGCOMM, Aug 2001.
[15] S. Ratnasamy, S. Shenker, and I. Stoica, "*Routing Algorithms for DHTs: Some Open Questions*". Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS), Mar 2004.
[16] S. Milgram, "*The small world problem* ". Psychology Today, 67(1), 1967.
[17] Jon Kleinberg, "*The small-world phenomenon: An algorithmic perspective* ". Proc. 32nd ACM Symp. on Theory of Computing (STOC), 2000.
[18] L. Barriere, P. Fraigniaud, E. Kranakis, and D. Krizanc, "*Efficient routing in networks with long range contacts*". Proc. 15th Intl. Symp. on Distributed Computing (DISC), 2001.
[19] G. S. Manku, M. Bawa and P. Raghavan, "*Symphony: Distributed Hashing in a Small World* ". Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS), Mar 2003.
[20] M. Castro, P. Druschel, Y. C. Hu, and A. I. T. Rowstron, "*Topology-aware routing in structured peer-to-peer overlay networks* ". Proc. Intl. Workshop on Future Directions in Distrib. Computing (FuDiCo), 2003.
[21] K. P. Gummadi, R. Gummadi, S. D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "*The impact of DHT routing geometry on resilience and proximity*". Proc. ACM SIGCOMM, 2003.
[22] H. Zhang, A. Goel, and R. Govindan, "*Incrementally improving lookup latency in distributed hash table systems*". In ACM SIGMETRICS, Jun 2003.
[23] G. S. Manku, "*Routing networks for distributed hash tables*". Proc. 22nd ACM Symp. on Principles of Distributed Computing (PODC), Jul 2003.
[24] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, "*Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*". In IEEE/ACM Transactions on Networking, Vol. 12, No. 2, pp. 205-218, Apr 2004.
[25] B. Y. Zhao, J. Kubiatowicz, A. Joseph, "*Tapestry: An infrastructure for fault-tolerant wide-area location and routing*". Tech. Rep. UCB/CSD-01-1141, Univ. of California at Berkeley, Computer Science Dpt, 2001.
[26] Xu, J., Kumar, A., and Yu, X., "*On the Fundamental Tradeoffs between Routing Table Size Network Diameter in Peer-to-Peer Networks*". In IEEE Jounal on Selected Areas in Communications, vol 22, no 1, Jan 2004.