Optimizing the finger table in Chord-like DHTs

Giovanni Chiola[1], Gennaro Cordasco[2], Luisa Gargano[2], Alberto Negro[2], and Vittorio Scarano[2]

[1]Università di Genova
Dip. di Informatica e Scienze dell'Informazione
16146 Genova - ITALY
chiolag@acm.org

[2]Università di Salerno
Dip. di Informatica e Applicazioni "R.M.Capocelli"
84081 Baronissi (SA) - ITALY
{cordasco,lg,alberto,vitsca}@dia.unisa.it

Abstract

The Chord protocol is the best known example of implementation of logarithmic complexity routing for structured peer-to-peer networks. Its routing algorithm, however, does not provide an optimal trade-off between resources exploited (the size of the "finger table") and performance (the average or worst-case number of hops to reach destination). Cordasco et al. showed that a finger table based on Fibonacci distances provides lower number of hops with fewer table entries. In this paper we generalize this result, showing how to construct an improved finger table when the objective is to reduce the number of hops, possibly at the expense of an increased size of the finger table. Our results can also be exploited to guarantee low routing time in case a fraction of nodes is assumed to fail.

Keywords:

Peer-to-peer overlay networks, routing efficiency and fault-tolerance, efficient protocol implementation, analytic evaluation, Montecarlo simulation.

1 Introduction and Background

Chord [8] is the best known, structured, peer-to-peer (P2P) protocol proposed in the literature. It implements the idea of "consistent hashing" [6] to build a Distributed Hash Table (DHT) based on several independent nodes.

Although the general idea behind Chord is very interesting and worth-while, the actual implementation of the lookup protocol [8] has some performance and reliability drawbacks. From a theoretical point view, for instance, Cordasco et al. [3] proposed a different way of constructing the so called "finger table" data structure, that provides a better trade-off between Diameter and Node Degree, thus leading to more efficient routing. From a practical point of view, the algorithm complexity analysis shows that "strong stabilization" in case of failure may require $O(N^3)$ steps [7], thus preventing the original protocol from properly scaling up in a realistic context where nodes are subject to failure.

In this paper we generalize the idea of Fibonacci distances by defining a family of functions that yield increasing finger tables which are always logarithmic in size, but that have higher constant factors. The idea is that the additional fingers yield increased routing performance in case all nodes are functional. Moreover, even in case a small fraction of nodes fails, thanks to the increased size of the finger table, the routing algorithm will degrade its performance more gracefully as compared to the original one. As shown by our analytical as well as simulation studies, the increased size of finger tables may substantially increase the reliability of Chord-like rings and reduce the expected number of hops even in case a non-negligible fraction of nodes fails.

The starting point for our work being F-Chord(1/2) [3], which provides the optimal trade-off between node degree and network diameter, our generalization keeps the minimality of the worst case number of hops, while the average number of hops is reduced by increasing the number of fingers. Our Montecarlo simulation results show that our generalization provides superior routing performance compared to two other alternatives already considered in literature, namely, Base-k (see section 2) and Extended Fibonacci [1].

Subsequently, Cordasco et al. [4] extended their neighbor-of-neighbor (NoN) $O(\log(N)/\log(\log(N)))$

This work has been financially supported by the Italian FIRB project Web-MiNDS `http://web-minds.consorzio-cini.it/`

routing algorithm to F-Chord(α). Indeed the very same extension applies to our generalized finger table as well, but this kind of extension is beyond the scope of this work [2, 5].

The rest of the paper is organized as follows. Section 2 analyzes the routing performance of overlays with finger tables characterized by Base-k distances (which include Chord as the special case $k = 2$). Section 3 re-derives the idea of F-Chord(1/2) routing table from a different perspectives, and then generalizes the idea to bases $k > 2$. Section 4 discusses the efficient implementation of the proposed finger table management protocol, providing details useful to interpret the simulation results reported in the following sections. Section 5 compares the routing performance estimates computed by Montecarlo simulation of our generalized tables against Base-k and extended Fibonacci finger tables. Section 6 assesses the problem of routing fault tolerance, presenting simulation results in case a fraction up to 10% of the nodes do not respond to routing requests. Finally, section 7 contains concluding remarks and discusses possible extensions of this work.

2 Base-k Finger Tables

The Finger Table in a Chord-like ring can be defined as an array of $\log(N)$ elements (where N is the total number of nodes in the overlay) that contain the addresses of some of the successors of the node along the ring at distances that increase as a function of the index in the table. In particular, if we denote by $J(i)$ the distance (also called jump) of the i-th finger from the considered node, in case of Chord we have: $J(i) = 2^i$
For example, if we consider a Chord ring containing N=16 nodes, each one characterized by a unique Id ranging from 0 to 15, each peer will build a finger table containing 4 elements, defined as: for all $n = 0, \dots, 15$:

$\text{fing}_n[0] = n + 1 \bmod 16 \quad \text{fing}_n[1] = n + 2 \bmod 16$
$\text{fing}_n[2] = n + 4 \bmod 16 \quad \text{fing}_n[3] = n + 8 \bmod 16$

In the worst case (e.g., when node 0 is requested to route to node 15), the number of hops required to reach destination, is equal to the size of the finger table, i.e., $\log_2(N)$. If we generate routing request at random, with uniform distribution from 0 to $N - 1$, the average number of hops is instead half the size of the finger table, namely $\log_2(N)/2$.

In analogy with the representation Base-k for natural numbers, if one wants to increase the size of the finger table in order to reduce the expected number of routing hops, a more or less obvious approach is to use a base greater than 2. For example one could construct a finger table whose jumps increase as a power of $k = 3$ rather than as a power of two, but that are repeated $k - 1$ times, i.e.: for all $l = 0, \dots, \log_3(N)$:

$$J((k-1)\cdot l) = k^l \quad J((k-1)\cdot l + 1) = 2\cdot k^l$$

For example, if we consider a Chord-like ring containing N=27 nodes, each one characterized by a unique Id ranging from 0 to 26, each peer will build a finger table containing 6 elements, defined as: for all $n = 0, \dots, 26$:

$\text{fing}_n[0] = n + 1 \bmod 27 \quad \text{fing}_n[1] = n + 2 \bmod 27$
$\text{fing}_n[2] = n + 3 \bmod 27 \quad \text{fing}_n[3] = n + 6 \bmod 27$
$\text{fing}_n[4] = n + 9 \bmod 27 \quad \text{fing}_n[5] = n + 18 \bmod 27$

Definition 2.1 *Base-k sequence of jumps.*
For any chosen base $k>2$, for all $l = 0, \dots, \log_k(N)-1$, for all $i = 0, \dots, k - 2$

$$J((k-1)\cdot l + i) = (i + 1)k^l$$

An overlay adopting this kind of finger table and adopting a Greedy routing algorithm is characterized by the following properties:

Property 2.1 *Node degree (i.e., size of the finger table) of the order of:*

$$(k-1)\cdot\log_k(N)$$

Property 2.2 *Diameter (i.e., worst case number of hops) of the order of:*

$$\log_k(N)$$

Proof : Consider two nodes s and t at distance d(s,t). Compute p such that $k^{p-1} \le d(s,t) < k^p$. Since in the interval $[k^{p-1}, k^p)$ there are $k - 1$ jumps at distance k^{p-1}, after one (Greedy) jump the distance reduces to less than k^{p-1}. By iterating, we have that after i jumps the distance reduces to less than k^{p-i}. Since $d(s,t) < N$, the diameter is $\log_k(N)$. □

Property 2.3 *Average number of hops (for uniformly distributed random routing requests) of the order of:* $\frac{k-1}{k}\cdot\log_k(N)$

Proof : Assume $N = k^p$ and consider a generic lookup request. Partition the routing in phases as follows: in phase i, the distance between the current source and the final target belongs to $[k^{i-1}, k^i)$. From the proof of Prop 2.2 we know that we need at most one jump to move from phase $i + 1$ to phase i. Moreover, after a jump from phase $i+1$, if the remaining distance is less than k^{i-1}, then we reach a phase $j < i$ without any jump in phase i. In particular, the probability that a jump of size k^{i-1} is needed is $\frac{k^i - k^{i-1}}{k^i} = \frac{k-1}{k}$. □

For small values of k, e.g. $k = 3$ or $k = 4$, this extension of the finger table may provide substantial performance improvements compared to the standard Base-2 tables, while increasing the size of the finger table by a relatively small multiplicative factor (1.5 in case $k = 4$). For larger values of k, instead, a diminishing return effect is expected, with almost linear increases of the finger table size (of the order of $k/\log(k)$) only partially compensated by very modest reductions of the path length (of the order of $1/\log(k)$).

3 Finger Tables based on Generalized F-Chord(1/2) Jumps

A first attempt to extend Fibonacci sequence of jumps in order to reduce the average number of hops at the expense of larger finger tables was presented in [1]. The idea was simply to reduce the speed of the increase of the jumps by using a recursive definition of the type: $J(i+1) = J(i) + J(i-k)$ with $k \geq 1$. This sequence was empirically studied by simulation and showed to be able to reduce the average path length for increasing values of k. Unfortunately the increase in size of the finger table prevented the use of large values for k.

In this paper, we present a completely different approach to the extension of Fibonacci type sequences of jumps. In this case we show by construction that the sequence of jumps is such that the minimal diameter is produced over the maximum range of ring size.

One of the distinctive characteristics of a finger table based on the Fibonacci sequence as the definition of the successive jumps $J(i)$ is that the difference between two consecutive jumps is the preceding jump, i.e.: $\forall i \in I\!N$ $J(i+2) - J(i+1) = J(i)$.
This relation has the following interesting consequence, when the finger table is adopted to support a Greedy routing algorithm.

Property 3.1 *Diameter with homogeneous Fibonacci jumps. If a routing request for node $v + x$ is issued to node x, and if $J(l+1) < v < J(l+2)$, then in one hop this request is forwarded to node $y = x + J(l+1)$, and on the latter node the following inequality holds: $w < J(l)$, where $w = v - J(l+1)$. As a consequence, the maximum number of hops will never exceed half the size of the finger table.*

This observation yields to the idea that half of the fingers can be removed from the table (either the ones with odd index or the ones with even index) without increasing the diameter of the overlay network. Taking only the odd (or even) numbers of a Fibonacci sequence as the jumps to define the finger table yields what in [3] was called F-Chord(α) in the particular case of $\alpha = 1/2$. This was proved to be the optimal trade-off between node degree (size of the finger table) and diameter (the worst case number of hops adopting the Greedy algorithm). Indeed in this case if a routing request for node $v + x$ is issued to node x, and if $J(l) < v < J(l+1)$, then in two hops this request is forwarded to node $y = x + J(l) + J(m)$ where $m \leq l$. In either case, on the latter node the following inequality holds: $w < J(l-1)$, where $w = v - J(l) - J(m)$.

Hence, the worst case number of hops is again equal to the size of finger table (as in Base-2, i.e., Chord), but in this case the size of the finger table is 28% smaller than Chord's.

Notice that this property is not enjoyed by Base-k jumps as defined in Section 2. In this section we generalize the property of F-Chord(1/2) to bases $k > 2$.

Definition 3.1 *Range of the ring with guaranteed diameter h, denoted $R(h)$. $R(h)$ is the maximum value $v \in I\!N$ such that $\forall w \in [0, v)$ a routing request for $x + w$ can be routed to destination in no more than h hops (i.e. $R(h)$ is the size of the greatest ring with diameter $R(h)$).*

For the sake of simplicity, let us consider first the case of a routing table derived by Base-3 jumps, as defined in 2.1. A random request can be routed in zero hops only if the request is for the node itself, hence $R(0) = 1$. On the other hand, a routing request can be routed in no more than one hop only if the request is for the node itself or for one of the nodes which Id is one of the fingers. If we consider the sequence of jumps for Base-3 finger tables, we can easily realize that $R(1) = 4 = k + 1$. This is simply because there is no jump in the finger table corresponding to the value $k + 1$.

Consider now the cases in which the request can be routed to destination in no more than 2 hops. We may observe that the two jumps following $k = 3$ are such that $J(3) - J(2) = J(4) - J(3) = 3 = R(1) - 1$. This is not optimal from the point of view of maximizing $R(2)$, as if for instance we have to route a value v such that $J(2) < v < J(3)$ we would forward the request to node $x + J(2)$ with a remaining displacement $w = v - J(2) < R(1) - 1$. Indeed, if we defined $J(3) = 7$ and $J(4) = 11$ (instead of 6 and 9), we would have obtained $R(2) = 15$ instead of 13. The improvement seems marginal in case of $h = 2$ (where the range is increased only by two compared to Base-3). However, the difference becomes substantial when we consider higher values of h.

In general, the jumps for the case $k \geq 2$ are recursively defined as follows.

Definition 3.2 *Generalized Base-k minimal diameter maximum range:*

$R(0) = 1$, $J(0) = 1$, *for each integer* $l \geq 0$,
for each $i = 1, \ldots, k-1$,

$$J((k-1) \cdot l + i) = J((k-1) \cdot l) + i \cdot R(l)$$

$$R(l+1) = J((k-1) \cdot l) + k \cdot R(l)$$

Notice that in the particular case $k = 2$, the sequence of jumps defined above is: 1, 2, 5, 13, 44, ..., which is the Fibonacci sequence without odd index numbers. Hence this is the proper generalization of F-Chord(1/2) to Base-k.

Now let us derive some interesting properties of the range function:

Property 3.2 *for each* $l \geq 1$,

$$R(l+1) = (k+1) \cdot R(l) - R(l-1)$$

Proof : By substitution of: $J((k-1) \cdot l) = J((k-1)(l-1)) + (k-1) \cdot R(l-1) = R(l) - R(l-1)$ in definition 3.2. □

Property 3.3 *for each* $l \geq 0$,

$$R(l) = \frac{\alpha^{l+1} - \beta^{l+1}}{\alpha - \beta} \qquad (1)$$

where $\alpha = \frac{(k+1)+\sqrt{(k+1)^2-4}}{2}$ *and* $\beta = \frac{(k+1)-\sqrt{(k+1)^2-4}}{2}$.

Proof : The property is the general solution of the linear recurrence relation of order 2, with initial values respectively 1 and $k+1$, where α and β are the two roots of the characteristic equation $x^2 - (k+1) \cdot x + 1$. □

Property 3.4 *Let d be the diameter of Base-k system with $N = R(d)$ nodes,*

$$\log_{k+\frac{k}{k+1}} \left(\left(\frac{k^2-1}{k^2} \right) \cdot N \right) < d < \log_{k+\frac{k-1}{k}} (N+1)$$

Proof : By (1) and observing that $\left(k + \frac{k-1}{k}\right) < \alpha < \left(k + \frac{k}{k+1}\right)$ and $\frac{1}{k+1} < \beta < \frac{1}{k}$, we have that $\left(k + \frac{k-1}{k}\right)^d - 1 < N < \left(\frac{k^2}{k^2-1}\right) \cdot \left(k + \frac{k}{k+1}\right)^d$. □

Property 3.5 *Let δ be the node degree of Base-k with $N = R(d)$ nodes,*

$$(k-1) \cdot \log_{k+\frac{k}{k+1}} \left(\left(\frac{k^2-1}{k^2} \right) \cdot N \right) + 1 < \delta$$

$$\delta < (k-1) \cdot \log_{k+\frac{k-1}{k}} (N+1) + 1$$

Proof : By definition 3.2 we have $\delta = (k-1)d + 1$. □

Property 3.6 *Let APL be the Average Path Lenght, i.e., the average number of hops (for uniformly distributed random routing requests) of a Base-k system with $N = R(d)$ nodes,*

$$\left(\frac{k-1}{k} \right) \cdot \log_{k+\frac{k}{k+1}} \left(\left(\frac{k^2-1}{k^2} \right) \cdot N \right) < APL$$

$$APL < \left(\frac{k+2}{k+3} \right) \cdot \log_{k+\frac{k-1}{k}} (N+1)$$

Proof : Consider a generic lookup request, and partition the routing in phases as follows: in phase i the distance between current source and final target is less than $R(i)$. If in phase i the distance is less than or equal to $J((k-1)(i-1))$, then we can move from phase i to phase $i-1$ without any jump. Therefore, at each phase i a jump is actually required to move to phase $i-1$ with probability at most $\frac{R(i)-J((k-1)(i-1))}{R(i)} = \frac{R(i)-R(i-1)+R(i-2)}{R(i)}$. Hence we have that

$$\begin{aligned} APL &< \sum_{i=1}^{d} \frac{R(i) - R(i-1) + R(i-2)}{R(i)} \\ &< \left(\frac{k+2}{k+3} \right) \cdot d \\ &< \left(\frac{k+2}{k+3} \right) \cdot \log_{k+\frac{k-1}{k}} (N+1) \end{aligned}$$

because of the UB in Prop. 3.4. By using the same argument and the corresponding lower bound in Prop. 3.4 we can easily show the lower bound on APL. □

In order to quantify the improvement in range introduced by this definition of jumps, let us consider the curves depicted in Figure 1. They show the growth of the size of the finger table as a function of the increase in the number of nodes in the overlay, simulated over the range from $N = 20$ up to $N = 900,000$. Notice that the curve corresponding to the generalized minimal diameter Base-k is not only lower than the one corresponding to Base-k, but also actually quite close to the one corresponding to Base-$(k-1)$, while the diameter is guaranteed by definition (see Def. 3.1) to be the same as the one of Base-k.

In Figure 1, the size of the finger table for the "extended Fibonacci" functions as proposed in [1] are also reported for the sake of comparison.

4 Finger table management in Chord-like protocols

As already pointed out by the authors [7], the finger table in a Chord-like routing protocol does not necessarily have to be up-to-date in order to guarantee

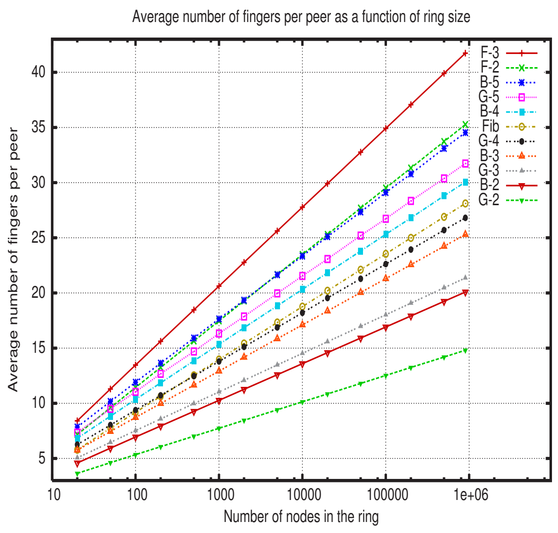


Figure 1. Number of elements in the finger table as a function of the number of nodes in the ring, where (as in the following figures) B-k denotes Base-k, G-k denotes generalized minimal diameter Base-k, Fib stands for Fibonacci, and F-k denotes "extended Fibonacci."

the correctness of the procedure. The only correctness requirement is the link to the next peer in the ring, namely the first entry in the finger table. An "error" in the first entry of the finger table may lead to the inability to properly route a message to destination, while an error in the subsequent entries of the finger table may simply slow-down the routing algorithm by unnecessarily increasing the number of hops needed to complete the route.

Therefore, our suggestion for managing the finger table is to only keep the first entry in the finger table constantly up-to-date by a proper synchronization protocol between any pairs of subsequent peers in the ring. When an anomaly is discovered in this link (namely during the addition of a new node or the removal of one node from the ring), that portion of the ring may be temporarily marked as "under repair" and any route request coming to the node that was responsible for that subset of keys may be delayed until the link is updated. By using this approach, one could remove the (global) "stabilization procedure" that was adopted in Chord. Such a stabilization procedure could affect availability as well as performance in case a relatively large fraction of nodes attach and/or detach themselves to/from the ring.

All subsequent entries in the finger table may be updated periodically with a periodicity that is selected as a functional parameter of the system. The addition/removal of a few nodes from the ring will change the actual distance of the peers that are linked in the finger table with respect to the "ideal" distance requested by the jump definition $J(i)$. However, such a temporary divergence will only (marginally) affect performance, and only until the entries of the finger table are recomputed. The choice of performing periodic updates of the finger table and allowing their entries to be slightly out-of-date has a beneficial effect on the requirements placed on the size of the finger table. In case all elements have to be kept up-to-date, the size of the finger table must be kept to a minimum in order to reduce the overhead associated with join/leave operations. On the other hand, if a divergence from the optimal is tolerated for some time, then the size of the finger table may grow without major impact on the ring maintenance protocol overhead.

A final implementation problem worth mentioning is related to the sparsity of the Ids associated with the peers compared to the Id space. Normally the space of node Ids is huge compared to actual size of the ring, so that the address space is almost empty. This fact, associated with the use of cryptographic Hash functions guarantees the absence of collisions in the Id space. However, this property may give rise to various implementations of the finger table starting from the same theoretical definition of jump sequences. In our simulation code we adopted the following choice to map the sequence of jumps into the actual (simulated) finger table each node. The definition of the jumps is mapped to the node Id space, and the mapped jump values are normalized by a multiplicative coefficient so that the range of the last normalized jump $R(h_{max})$ is equal to the highest Id in the Id space. Then a lookup operation is simulated in order to identify the peer that is responsible for the given normalized jump value, and the address of this peer is adopted as the i-th entry in the finger table.

The filling of the finger table starts with the highest index jump, and continues until the immediate successor of the current node is inserted in the table, thus automatically adjusting to various dimensions of the ring. Due to the random assignment of Ids to peers, the finger table of one particular node has not necessarily the same size of another node in the same ring. The actual number of fingers for a particular node depends on the actual distribution of Ids of its neighbors. For this reason, the size of the finger table may only be defined in a statistical way, by computing the average over all peers of the actual size of their individual finger tables. This explains why we referred to the size of the finger table as the “average number of fingers per peer” in Figure 1.

5 Montecarlo Simulation of the Greedy routing algorithm

In order to assess the effectiveness of our proposed finger tables, we used a very simple Montecarlo simulation approach. A virtual ring is constructed by randomly generating the IDs associated with the prescribed number of nodes. Then the finger list for each node is constructed according to the chosen distance function. Finally, a large number of random, uniformly distributed, independent routing requests are issued to the node with the lowest ID in the ring, and routed through the nodes simulating the Greedy routing protocol. The number of hops is counted for each route, and statistics are collected. 99% confidence level intervals are estimated in order to ensure the precision of the simulated results. The experiments are repeated until the confidence intervals become small enough. All the results reported in the paper have 3 digits precision with 99% confidence level.

The results obtained by our simulation experiments are depicted in Figure 2. Both average values and 95 percentile are depicted, the latter being defined as the least integer value that is not exceeded in at least 95% of the routing requests. As one can see from the diagrams, G-2 (which is F-Chord(1/2)) yields substantially worse performance than B-2, due to its substantial reduction in the size of the finger table. However, as we increase the base to values $k > 2$, we can notice that the performance gap of G-k with respect to B-k vanishes, with G-3 already quite close to B-3 even in case of large rings (where G-3 uses quite less fingers than B-3). The comparison against (extended) Fibonacci shows that G-4 is superior to Fibonacci and G-5 is superior to F-2, in spite of the quite lower number of fingers in case of large rings.

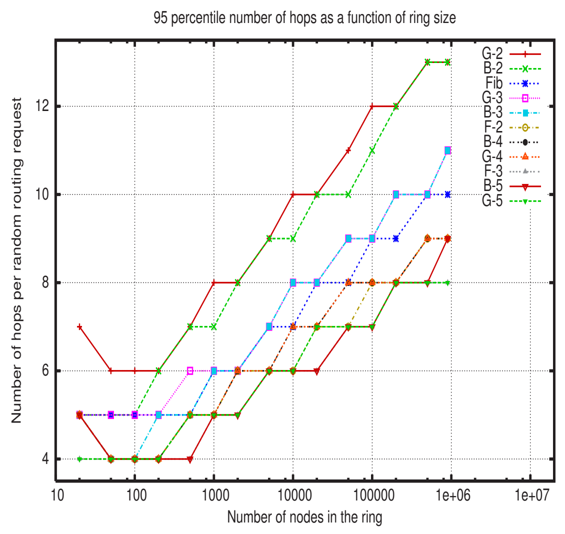
In order to better appreciate the efficiency of the various routing tables, we defined a “weighted routing cost” as follows:

$$wcost = 0.4 \cdot \text{no\_fingers} + 0.3 \cdot \text{avg\_hops} + 0.3 \cdot \text{p95\_hops}$$

All parameters are to be minimized in order to obtain good performance with minimal resources, so that the lower the weighted cost the better. The results obtained by our simulation experiments are depicted in Figure 3. As one can see, the cost of “extended Fibonacci” is definitely higher than the others. Notice also that the cost of G-$(k+1)$ is roughly comparable to the one of B-k, thus suggesting the possibility of obtaining higher performance at similar cost.

6 Montecarlo Simulation of rings with failures

One possible drawback of our proposed finger table management algorithm with lazy update of the finger tables is the possibility that one (or more) links may point to peers that disconnected from the ring (or failed) after the last link update. In order to deal with this case, the routing algorithm must be prepared to interact with peers that are not ready to communicate. Notice, however, that the first finger (i.e. the immediate successor) is assumed to be constantly updated,



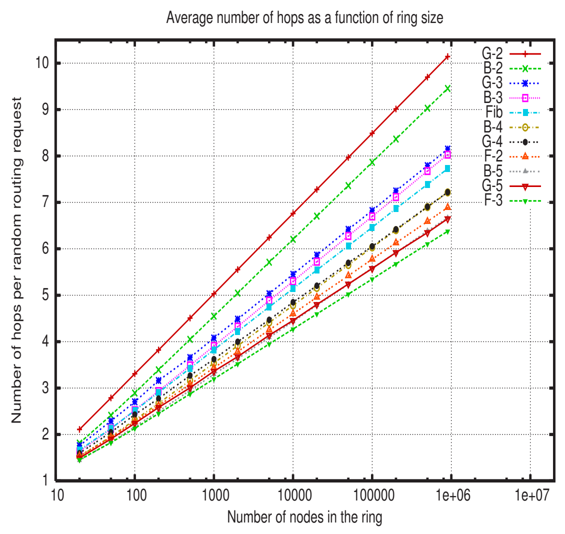


Figure 2. Average and 95 percentile number of hops for random routing requests as a function of the number of nodes in the ring, for the considered finger tables.

so that each searched key is always associated with a functioning node, and no lookup will ever fail: the only consequence of a non up-to-date finger table is a possible slow-down of the lookup itself, with higher number of hops as compared to the "normal case" studied in the previous section.

Our proposed solution is to set up a time-out after forwarding a routing request to a peer and reset it upon receipt of acknowledgment. If no acknowledgment is received before the time-out elapses, then the contacted peer is assumed not to belong to the ring, and the routing request is forwarded to the peer whose address is immediately before the failed one in the finger table (i.e. the one at a distance immediately shorter than the "optimal" one that did not respond). In case of multiple failures, the node will keep trying closer and closer neighbors, possibly until the first element in the finger table is reached (remember that this link is assumed to be kept constantly up-to-date, so that progress of at least one step in the ring is always guaranteed, unless this portion of the ring is momentarily under repair).

In order to obtain an optimistic bound on the routing time in presence of failures, in our simulations we assumed that the time-out is set to only twice the average hop time (this is the average delay for an ACK message to come back to the sender in case the peer is up and running). This timeout is added each time the simulated routing algorithm makes an attempt to forward a request to a non operational node.

The results obtained by our simulation experiments are depicted in Figure 4, for varying fractions of failed nodes ranging from 0 to 10%. The most interesting observation that one can make is that the G-k family of finger tables is less sensitive to failures than both B-k and (extended) Fibonacci. This is probably due to the more efficient use of the fingers that is made, which on one hand reduces the number of hops (hence reducing the probability of getting a failed node in the route), and on the other hand reduces the size of the finger table (thus reducing the total number of failed fingers).

The same kind of benefits can be also appreciated in the diagrams of the weighted costs, depicted in Figure 5. G-3 and G-4 appear to offer the most stable cost as a function of node failures.

7 Conclusions

In this paper we have presented the extension of the F-Chord(1/2) finger table to base $k \geq 2$. We studied the characteristics of the family of functions both analytically and by Montecarlo simulation. The parameter k may be increased in order to reduce average and maximum number of hops, at the expense of an increased size of the finger table for a given number of nodes in the network.

We defined a concept of weighted routing cost that takes into account average and probability distribution of the number of hops as well as the number of fingers

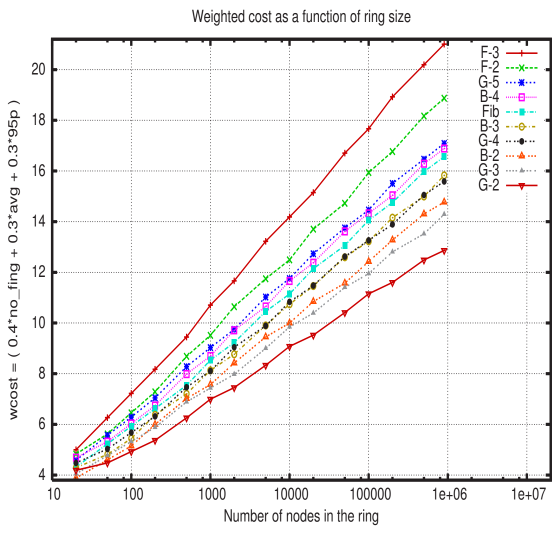


Figure 3. Weighted routing cost as a function of the number of nodes in the ring.

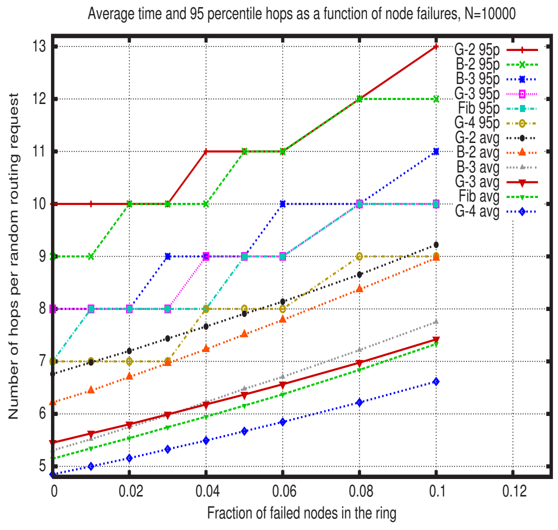


Figure 4. Average and 95 percentile number of hops for random routing requests in a 10,000 node ring as a function of the fraction of nodes that failed.

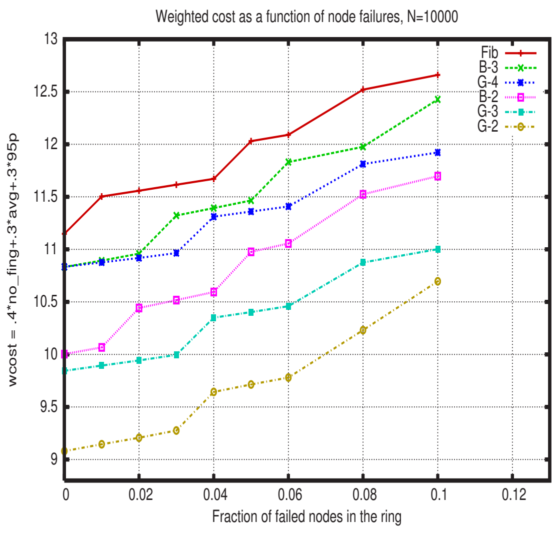


Figure 5. Weighted cost in a 10,000 node ring as a function of the failure ratio.

in the table. According to this weighted cost definition, our proposed finger table makes optimal use of the available links by extending the range of the ring size as compared to Base-k, without losing performance in case all nodes in the ring are assumed to be functional. Moreover, our new finger table exhibits reduced sensitivity to failures.

An interesting development of this work is the application of our proposed jump functions to pseudo-randomized finger tables associated with a neighbor-of-neighbor (NoN) routing algorithm, as already done in [4] for the case of F-Chord(α). Even in this case a predefined increase in the size of the finger table should further improve performance in normal cases and should reduce the performance degradation in case a fraction of the nodes fails.

References

[1] G. Chiola. "Extended Fibonacci Distances for Fault-Tolerant Routing in Chord-Like DHTs". In *Proceedings of first International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P '04) (in conjunction with MASCOTS 2004) Volendam, The Nederlands*, pages 10–15. IEEE Computer Society, October 2004.

[2] G. Chiola, G. Cordasco, L. Gargano, A. Negro, and V. Scarano. "Overlay networks with class". In *Proceedings of 8th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2005) Las Vegas, Nevada, USA*. IEEE Computer Society, December 2005.

[3] G. Cordasco, L. Gargano, M. Hammar, A. Negro, and V. Scarano. "F-Chord: Improved Uniform Routing on Chord". In *Proceedings of 11th Colloquium on Structural Information and Communication Complexity (Sirocco '04), Smolenice Castle, Slovakia*. Springer-Verlag Berlin Heidelberg New York, June 2004.

[4] G. Cordasco, L. Gargano, M. Hammar, A. Negro, and V. Scarano. "Non-uniform deterministic routing on F-Chord(α)". In *Proceedings of first International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P '04) (in conjunction with MASCOTS 2004) Volendam, The Nederlands*, pages 16–21. IEEE Computer Society, October 2004.

[5] G. Cordasco, L. Gargano, M. Hammar, and V. Scarano. "Degree-Optimal Deterministic Routing for P2P Systems". In *Proceedings of 10th IEEE Symposium on computers and communications (ISCC '05) La Manga del Mar Menor, Cartagena, SPAIN*, pages 158–163. IEEE Computer Society, June 2005.

[6] D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web". In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, ACM Press, El Paso, TX, USA*, pages 654–663, 1997.

[7] I. Stoica, R. Morris, D. Karger, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications". In *Technical Report TR-819, MIT, LCS*, April 2001.

[8] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications". In *IEEE/ACM Transactions on Networking (TON), Volume 11, No. 1*, pages 17–32, February 2003.