

# Relaxed-2-Chord: Efficiency, Flexibility and provable Stretch

Gennaro Cordasco\*  
University of Salerno  
cordasco@dia.unisa.it

Francesca Della Corte\*  
University of Salerno,  
francescadellacorte@virgilio.it

Alberto Negro\*  
University of Salerno  
alberto@dia.unisa.it

Alessandra Sala†  
University of California at Santa Barbara  
alessandra@cs.ucsb.edu

Vittorio Scarano†  
University of Salerno  
vitsca@dia.unisa.it

**Abstract.** Several proposals have been presented to supplement the traditional measure of routing efficiency in P2P networks, i.e. the (average) number of hops for lookup operations, with measures of the latency incurred in the underlying network. So far, no solution has been presented to this “latency” problem without incurring in extra and heavy management costs. We propose Relaxed-2-Chord, a new design of the traditional Chord protocol, that is able to fit the routing tables with low latency nodes, doing a parasitic measurement of nodes’ latency without adding any overhead. The solution that we present is a Distributed Hash Table system whose aim is to combine the routing efficiency and flexibility of the Chord protocol – i.e. a good degree/diameter tradeoff - and a provable optimal hop by hop latency. Our work is inspired by the recent Lookup-parasitic random sampling (LPRS) strategies which allow to improve the network stretch, that is, the ratio between the latency of two nodes on the overlay network and the unicast latency between those nodes. Relaxed-2-Chord reaches the same results as LPRS without introducing any overhead.

## 1. Introduction

Over the last eight years, Peer-to-peer (P2P) systems quickly became very popular. Several are the applications based on the P2P paradigm that offer disparate services ranging from file-sharing to grid computing, through distributed file storage and collaborative systems. Traditional P2P systems are completely distributed and use a scalable Distributed Hash Table (DHT) as a sublayer. A DHT is a self-organizing overlay network that allows adding, deleting and looking up hash table elements. Several protocols [18, 7, 20, 24] have been proposed which present a struc-

tured overlay network such that lookups require a small number of hops, typically  $O(\log n)$  overlay hops in average, where  $n$  is the network size.

However, the traditional measure of routing efficiency, that is the average number of hops for a lookup request, must be supplemented with the consideration that the routing latency between two nodes on the overlay network can be different from the unicast latency between those two nodes on the underlying network. When the main goal of a P2P system is to design a network with “low latency” characteristics, then the following three aspects play the main role:

- Decrease the path length: by Little’s Law [14], the reduction in the path length results in significant reduction in the overall network traffic. Systems like the one described in [13] have proved the low latency benefits that come from the reduction of the paths length.
- Load balancing: systems that seek to force the traffic through low latency paths could produce an unbalanced load on the network that yields the degradation of the latency on the most used paths.
- Decrease latency hop by hop: although the path length is an important metric for measuring efficiency of the routing, it does not adequately address the issue of end-to-end latency because each overlay hop could potentially involve significant delays (intercontinental links, satellite links, etc.).

Two main strategies have been employed, in the last years, to solve this problem. The first one is to give more flexibility in the construction of routing tables such that, through sampling strategies, the nodes in the routing table are continuously updated making the choice respect their proximity, assuming that high proximity means low latency. The second one is based on using synthetic coordinates to predict inter-node latencies, as shown in [17]. Unfortunately these solu-

\*ISISLab-Dipartimento di Informatica e Applicazioni, Università degli Studi di Salerno, Via Ponte don Melillo, Fisciano (SA) 84084, ITALY

†Computer Science Department, University of California at Santa Barbara, USA

tions, which potentially work very well, are difficult to maintain/implement and make load balancing hard to support.

From a theoretical point of view it is significant to prove an optimal *latency stretch* (hereafter *stretch*) of the network. The stretch of a network is the ratio between the latency of two nodes on the overlay network and the unicast latency between those nodes.

Traditional DHTs provide a large stretch, since the overlay network topology bears no resemblance with the underlying network. Large stretches results in poor DHT performance even if the routing algorithm adopted by the DHT is optimal (compared to the number of hops on the overlay network).

Today, applications like “video on demand”, “Voice-over-IP”, “Video Conferencing” and “Gaming” require reliable and low latency networks. These requirements have implied that the “veteran” client-server architectures are surrendering to the Peer-to-peer one [21, 11]. Indeed, Peer-to-peer architectures are becoming more and more used from these applications such as massively-multiplayer online games (MMOGs) [11], where the players can be connected - end to end - with each other. Improve the latency on the - end to end - connections is one of the main challenges in the real-time applications.

Our work is motivated by the challenge to reach an optimal stretch on the paths without adding extra overhead such that real-time applications can fulfill their low latency requirements. This paper emphasizes the capability of building low latency paths without adding extra costs which makes our protocol eligible to a real implementation.

The choice to propose a modified version of the well known “Chord” protocol is motivated by (a) its simplicity and, yet, efficiency, (b) by the large corpus of research on Chord, and (c) by the informal feeling that the ring is a simple structure that can greatly benefit from improvements on stretch, further supported by Gummadi et al. [9] remarks that state: *“Our basic finding is that, despite our initial preference for more complex geometries, the ring geometry allows the greatest flexibility, and hence achieves the best resilience and proximity performance”*.

### 1.1. Our Results

Our work seeks to prove the goodness of a parasitic sampling strategy on Relaxed-2-Chord a bidirectional and more flexible version of Chord. In the remaining of this paper we show how to improve the latency on the routing paths during the lookup operations and without any extra cost. We propose a simple and effective technique where a node can gain information on the latency to other nodes in the network during the normal lookup operations and modifies its routing table according to the latency requirements.

## 2 Related Works

Many different strategies have been developed in the last few years in order to improve the latency in Peer-to-peer systems.

### IDs based on location systems.

The basic problem of modeling Internet topology as a geometric space (e.g. a 3-dimensional Euclidean space) has already been approached in systems like [17]. The main idea of these systems is to use a well-defined coordinate system and a corresponding well-defined distance function, and characterize the position of any host in the Internet by a point in this space where the distances between any two hosts can be predicted with high accuracy by the distance function evaluated on the hosts coordinates. GNP [17], for example, demonstrated that it is possible to calculate synthetic coordinates, and that they can be used to predict Internet latencies. GNP relies on a small number (5-20) of landmark nodes; other nodes measure latency to the landmarks to help them choose coordinates. The choice of the nodes to be used as landmarks can significantly affect the accuracy of latency predictions made by GNP, but the main drawback is that GNP is not self-organizing because of the need to fix/manage the landmarks. Other protocols like Vivaldi [4] compute synthetic coordinate without the support of Landmark nodes. Indeed, in Vivaldi, each node computes coordinates for itself. Each time a node communicates with another node, it measures the latency to that node, and then adjusts its coordinates to minimize the error between measured latencies and predicted latencies. Although Vivaldi seems to be working better than GNP, it does not solve the unbalanced nature of “proximity” network that can cause a non-uniform distribution of nodes in the key space, leading to performance reduction.

### Latency-aware DHT systems.

DHT systems show optimal bound in lookup performance and so several approaches have been proposed for designing DHT systems with low latency[3]: (a) PROXIMITY IDENTIFIER SELECTION where the nodes ID are assigned in order to ensure that nodes close in the underlying network are also close in the metric space; An example of Proximity identifier selection was proposed to improve routing performance in CAN [19], where each node measures its round-trip delay to a set of pre-determined landmarks, and accordingly places itself in the key space. With  $t$  such landmarks,  $t!$  ordering are possible, so the key space is divided into  $t!$  equal-sized partitions, and each node will join the portion that matches its landmark ordering. Unfortunately this approach leads to an unbalanced distribution of nodes into the key space. (b) PROXIMITY ROUTING SELECTION proposes

a lookup algorithm which also investigates the underlying topology. This approach trades off the number of overlay hops per path and the network distance traversed by each hop. (c) PROXIMITY NEIGHBOR SELECTION constructs the routing tables for each nodes taking into account the network proximity among nodes. Neighbors are chosen to refer to nodes that are nearby in the network topology. In this way, the distance traveled by messages can be minimized without a significant increase in the number of routing hops.

### Lookup-parasitic random sampling (LPRS).

In [23] the authors propose a parasitic strategy which allows to reduce the stretch without altering the routing scheme. Such strategy can be applied to several geometric routing DHT schemes (i.e., where the routing, at each step, forwards the message to a peer that is closer to the destination peer). The LPRS strategy follows the proximity neighbor selection approach and, in particular, it provides a simple way to find a set of good neighbors exploiting a sampling strategy. Sampling nodes are exchanged during the lookup process. In particular each lookup message contains additional information (i.e. IP addresses) about a set of nodes encountered during the lookup process. Thereafter, the source node is able to sample<sup>1</sup> all the nodes in this set.

The authors argue that the Lookup-parasitic random sampling can be done without extra-cost, since the effort can be piggybacked over normal DHT functionality, however each sample requires several operations (e.g., pings) to get a reasonable estimate of the latency, and hence a certain overhead is introduced.

In [23] the authors analyzed also several sampling strategies and observed that using a bidirectional routing scheme it is possible to use further sampling strategies which they leave for further investigations.

Some other systems have been proposed: Tulip [1] and LAND [2] achieve provably low stretches, but lack the stability property, that is, those systems are not robust to frequent peer arrivals, departures and fails; eQuus [15] builds an overlay that comprises both provable fault-tolerance and provable locality-awareness, even if it lacks load balancing property when the underlying network results clustered in unbalanced manner.

## 3. Preliminaries

In this section we analyze several variants of Chord that are of interest for the rest of this paper. We consider a set  $N$  of  $n$  nodes lying on a ring of  $2^m$  identifiers (labeled from 0 to  $2^m - 1$  in clockwise order).

<sup>1</sup>When we say that a node  $v$  samples a node  $u$ , we mean that  $v$  measures the latency to  $u$  and depending on the measured value,  $v$  may update its neighbors set.

$u, v, z$	Nodes in the Network
$n$	Number of nodes in the network
$m$	Number of bits to identify a node
$N_\delta(v)$	Number of nodes within latency $\delta$ from $v$
$\ell(u, v)$	Latency between $u$ and $v$

**Table 1.** Notations used

### Chord[20].

Chord provides a hash-table functionality by mapping  $n$  nodes (using consistent hashing [12]) to identities of  $m$  bits placed on a ring network. Each node  $v$  is connected<sup>2</sup> with its predecessor and its successor on the ring. Consistent hashing ensures that resource keys and node identifiers are spread roughly uniformly in the key space, ensuring approximately balanced load in the network.

In order to achieve fault tolerance, each peer  $v$  maintains a *successors list* which consists of the next  $f$  *successor* peers. Successors list allows to keep correct the ring even if many *successor* turn out to point to crashed peers. The only situation in which Chord cannot guarantee finding the current live successor to a key is if all  $f$  peer's *successor* fail simultaneously. Relatively small values of  $f$  (such as  $\Omega(\log n)$ ) makes the probability of simultaneous failure vanishingly small.

Each peer  $v$  also maintains a *routing table* (a.k.a., *finger table*) with  $O(\log n)$  entries, spaced exponentially around the key space, such that the  $i^{\text{th}}$  entry stores the identity of the *responsible* of  $v + 2^{i-1}$  on the identifier ring. Routing efficiency is achieved by using the *finger table*. An example of these pointers is shown in Figure 1. We now provide the formal definition of link connections<sup>3</sup> of this network:

**Chord [20]:** For each  $0 \leq i < m$ , node  $v$  is connected by edges to the nodes<sup>4</sup>  $v + 2^i$ .

The main emphasis in Chord's design is robustness and correctness, achieved by using simple algorithms with provable properties even under concurrent joins, leaves and failures.

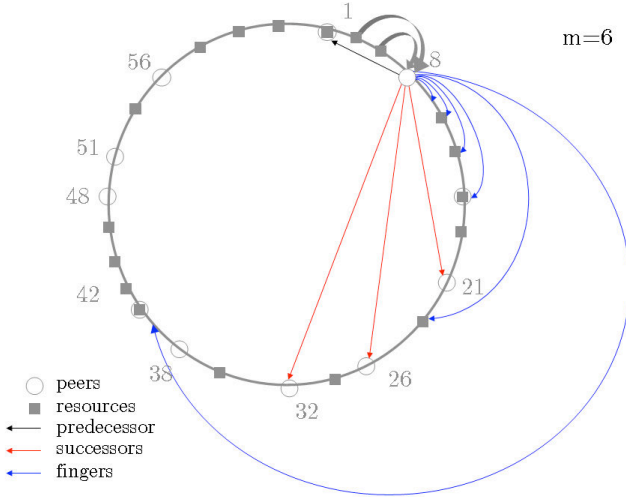
### 2-Chord [6].

A Chord-like bidirectional routing scheme, called 2-Chord, has been proposed in [6]. This protocol proposes a new stabilization procedure that does not need any periodic update (such as Chord fix.finger), with no harm to the efficiency

<sup>2</sup>When we say that a node  $v$  is connected to  $u$ , we mean that  $v$  is connected to  $u$  if  $u$  is a node, otherwise  $v$  is connected to the first node that follows  $u$ .

<sup>3</sup>We assume however that each node always maintains its predecessor and successor in the ring.

<sup>4</sup>All the arithmetic operation on the ring are mod  $2^m$ .



**Figure 1. An example of Chord ring: The Figure depicts a ring with 11 peers and 16 resources and shows, in particular, connections and resources maintained by peer 8.**

and simplicity that are some of the many good characteristics that made Chord a popular choice. Formally we can define the link connections as the following:

**2-Chord** [6]: For each  $0 \leq i < m - 1$ , node  $v$  is connected by edges to the nodes  $v + 2^i$  and  $v - 2^i$ .

#### R-Chord [16].

Recently, some of Chord's results were improved by introducing both randomization in the choice of each link to be established and a novel routing strategy, called NoN (Neighbors of Neighbors) greedy routing. The formal representation of the overlay connections among nodes in this network can be summarized as the following:

**R-Chord** [16]: For each  $0 \leq i < m$ , let  $R(i)$  denote an integer chosen uniformly at random from the interval  $[0, 2^i)$ , node  $v$  is connected by edges to the nodes  $v + 2^i + R(i)$ .

#### $d$ -powerlaw latency expansion.

Let  $N_\delta(v)$  denote the number of nodes in the network that are within latency  $\delta$  from  $v$ . Informally, for  $d \geq 1$ , a graph  $G$  has a  $d$ -powerlaw latency expansion if  $N_\delta(v)$  grows (i.e., "expands") proportionally to  $\delta^d$ , for all nodes  $v \in G$ . Some simple examples of graph families with power-law latency expansion are rings ( $d = 1$ ), lines ( $d = 1$ ), and meshes ( $d = 2$ ).

Let  $\Delta = \max_{u,v \in G} \ell(u, v)$ , where  $\ell()$  denotes the latency function, denote the latency diameter of  $G$ .

## 4. Network Design: Relaxed-2-Chord

A careful analysis of the results reviewed in Section 2 shows that the key idea to improve the latency in a system is to make the overlay network topology less rigid. In order to build such a topology, we first need a more flexible routing scheme. Therefore, we allow more than one choice for each link, that needs to be established during the routing table construction, by expanding the network portion where a node can select its routing table entries.

On the other hand, it is fundamental to have an effective and efficient sampling scheme which is able to select low latency nodes from any starting node avoiding communication overhead. Since bidirectional routing schemes guarantee a large number of sampling strategies, in this paper we explore how to reuse the bidirectional characteristic of 2-Chord [6].

We introduce Relaxed-2-Chord which is a hybrid protocol that benefits from the bidirectional structure explored in 2-Chord [6] and the randomized scheme implemented in R-Chord.

Relaxed-2-Chord profits from its bidirectional structure because during the routing, if the previous hop went clockwise (resp. counterclockwise) the current node can easily *sample* a link which goes counterclockwise (resp. clockwise). Moreover, Relaxed-2-Chord benefits from its flexible routing table because each entry results to be the most low latency node in that portion of the network.

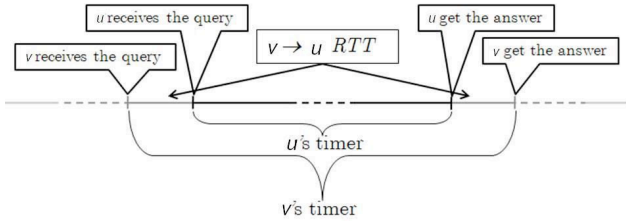
Summarizing, we propose Relaxed-2-Chord which is able to obtain a provable stretch with no extra-cost, meanwhile showing its efficiency and simplicity as every Chord-like network.

In the following sections we formally describe how to achieve flexibility in the lookups and how to improve latency in the overlay paths. First we present the strategy used to connect nodes among each other, then we examine the routing algorithm and finally we describe the sampling method.

### 4.1 Overlay Network Structure

In this section we describe the strategy used to connect nodes among each other in the overlay network. With the following strategy we are able to guarantee efficient routing (i.e. asymptotically bounded by  $O(\log n)$  overlay hops) and a provable stretch.

**Definition 1 (Relaxed-2-Chord:)** For each  $0 \leq i < m - 1$ , node  $v$  is connected to the closest node, in terms of latency, which belongs to the interval  $[v + 2^i, v + 2^{i+1})$  and to the closest node, in terms of latency, which belong to the interval  $[v - 2^{i+1}, v - 2^i)$ .



**Figure 2. RTT estimation, during a recursive lookup, through timers.**

In details, each node  $v$  maintains a routing table with up to  $2m - 2$  entries: clockwise connections are called *forward fingers* (the  $i^{\text{th}}$  forward finger belongs to  $[v + 2^{i-1}, v + 2^i)$ ), while counterclockwise connections are called *back fingers* (the  $i^{\text{th}}$  back finger belongs to  $[v - 2^i, v - 2^{i-1})$ ).

We remark that using the same argument of [20], it is simple to prove that every node in Relaxed-2-Chord has  $O(\log n)$  fingers with high probability (w.h.p.). Note that, if a node  $v$  is connected to a node  $u$  and, in particular,  $u$  is the  $i^{\text{th}}$   $v$ 's forward (resp. back) finger then  $u$  can sample  $v$  and consequently may update its  $i^{\text{th}}$  back (resp. forward) finger.

We notice also that nodes' latencies affect only the finger table which allows us to implement low latency lookup operations. In order to guarantee both load balancing and fault tolerance, either the identifier assignment function or the successors list are completely independent from nodes' latency and randomized using a standard approach (that is, via a consistent hashing function.) In this way, our system can easily manage the problem of correlated node failure. Indeed, even assuming that a big portion of the network fails simultaneously, the probability that a node loses all its successors remains sensibly small.

Our effort is to develop a sampling strategy without any extra-cost. In order to do that, we present a strategy that is completely "parasitic" and does not require any additional communication. Indeed, latency estimations are obtained during the normal DHT functionality with no harm to the efficiency of the protocol, as we will formally describe in Section 4.3.

In the following we assume that each node  $v$  maintains, in the routing table, for each neighbor  $u$  the value of the latency for the connection between  $v$  and  $u$  ( $v.\text{latency}[u]$ ). This estimation is continuously updated according to a timer which allows to measure the round-trip time for each query. We remark that our approach does not need synchronized timers. The nodes should only agree on the metric used for measuring the time. Indeed each node  $v$  estimates the round trip time (RTT) to a node  $u$  each time it forwards a query to  $u$ . The estimation is obtained by subtracting the

time spent by  $u$  to get the answer from the time spent by  $v$  to get the answer (cf. Fig. 2).

## 4.2 Routing in Relaxed-2-Chord

The routing is performed on Relaxed-2-Chord following a greedy scheme. Indeed, in contrast with the Chord protocol, since 2-Chord is bidirectional, overshooting is allowed. Hence, the distance  $d(a, b)$  between any two IDs, for example  $a$  and  $b$ , is defined as:

$$d(a, b) \stackrel{\text{def}}{=} \min\{(a - b), (2^m - (a - b))\}. \quad (1)$$

Therefore,  $d(a, b) = d(b, a)$ . Consider a lookup operation starting at a node  $v_0$  for a generic key  $k$  which belongs to the node  $v_t$ . For each step  $i \in [0, t - 1]$  the node  $v_i$  forwards the message to its neighbor node  $v_{i+1}$  that is the closest to the destination with respect to the metric distance  $d()$ .

Our strategy, as any proximity neighbor selection approach (cf. Section 2), requires recursive routing, rather than iterative. Indeed, while in the recursive approach nodes communicate only with their neighbors, in the iterative approach, nodes communicate with their neighbors only during the first routing step. This can invalidate any proximity neighbor selection strategy since the latency is clearly not optimized between non-neighbor nodes. In Section 5 we will formally prove the asymptotical bound on the number of hops (i.e  $O(\log n)$ ) during the lookup operations and then we will validate our results via simulations.

## 4.3 Random Sampling in Relaxed-2-Chord

The random sampling strategy, which allows to update estimated latencies, is described in Figure 3 compared with the standard lookup on Chord that is given on the left such that it is easy to substantiate our no-extra-cost statement.

In order to explain how the random sampling works in Relaxed-2-Chord, let's consider a generic lookup step, from node  $v_i$  to  $v_{i+1}$ . Since the mapping of resources to the identifier space is pseudo-random (that is, it is based on a consistent hashing function) the node  $v_{i+1}$  can consider  $v_i$  as a random sample. In particular, by Definition 1, if  $v_{i+1}$  is the  $j^{\text{th}}$   $v_i$ 's forward (resp. back) finger then  $v_i$  is a feasible  $j^{\text{th}}$  back (resp. forward) finger for  $v_{i+1}$ . Thus, when  $v_i$  forwards a query to a node  $v_{i+1}$  it also sends its estimated latency to  $v_{i+1}$  ( $v_i.\text{latency}[v_{i+1}]$ , see also Figure 3 (right) line 7). In this way,  $v_{i+1}$  can easily sample  $v_i$  and, where appropriate, can update its finger table entry (cf. Figure 3 (right) line 2, where  $r$  and  $\ell$  represent respectively the sampled node and its estimated latency).

Note that, due to routing asymmetry, it is well known that latency estimation are not symmetric (i.e.  $\ell(v, u) \neq$

<pre> <b>s.lookup</b>(<i>id</i>) 1: <b>if</b> <i>id</i> ∈ (<i>s</i>, <i>s</i>.successor] <b>then</b> 2:   <b>return</b> <i>s</i>.successor 3: <b>else</b> 4:   <i>t</i> = <i>s</i>.closest_preceding_node(<i>id</i>) 5:   <b>return</b> <i>t</i>.lookup(<i>id</i>) 6: <b>end if</b> </pre>	<pre> <b>s.lookup</b>(<i>id</i>, <i>r</i>, <i>ℓ</i>) 1: timer.start() 2: <i>s</i>.sample(<i>r</i>, <i>ℓ</i>) 3: <b>if</b> <i>id</i> ∈ (<i>s</i>, <i>s</i>.successor] <b>then</b> 4:   <b>return</b>(timer.value(), <i>s</i>.successor) 5: <b>else</b> 6:   <i>t</i> = <i>s</i>.closest_node(<i>id</i>) 7:   (<i>ct</i>, <i>target</i>) = <i>t</i>.lookup(<i>id</i>, <i>s</i>, <i>s</i>.latency[<i>t</i>]) 8:   <i>s</i>.update(<i>s</i>.latency[<i>t</i>], (timer.value() − <i>ct</i>)/2) 9:   <b>return</b>(timer.value(), <i>target</i>) 10: <b>end if</b> </pre>
--	---

**Figure 3. Lookup on Chord (left) and Relaxed-2-Chord (right).**

$\ell(u, v)$ ). However, we are not interested in very accurate values but we are rather looking for a coarse-grained latency distinction and our methodology is appropriate for that purpose.

Each node  $v_i$  on the path uses a local timer in order to measure the time required to get the answer for a particular query (see Figure 3 (right) line 1). Thereafter, when the node  $v_i$  is able to answer a query, the current value of its timer, which represent the time spent by  $v_i$  to obtain the answer, is returned, together with the answer, to the preceding node on the path,  $v_{i-1}$  (Figure 3 (right) line 4 and 9). In this way,  $v_{i-1}$  can extrapolate its latency to  $v_i$  ( $(\text{timer.value()} - ct)/2$ , where  $\text{timer.value()}$  is the time spent by  $v_{i-1}$  to obtain the answer and  $ct$  is the time spent by  $v_i$  to obtain the answer, see Figure 2 too). Thus,  $v_{i-1}$  is able to update its estimated latency to  $v_i$  (Figure 3 (right) line 8). Finally, the procedure `closest_node(id)` differs from `closest_preceding_node(id)` because in our routing strategy overshooting is allowed, hence the procedure `closest_node(id)` finds the neighbor node that is closer to the destination according to the metric distance  $d()$  presented in Definition (1).

We stress that no sampling LPRS-style, i.e. with ping, is performed in our solution.

## 5. Efficiency and Latency Analysis

In this Section we prove the asymptotical bound on the number of hops in the lookup operation, the expected latency of a lookup and the results that come from the simulations done.

**Theorem 1** *The number of nodes that must be contacted to resolve a lookup query in an  $n$ -node Relaxed-2-Chord networks is  $O(\log n)$  w.h.p.*

*Proof.* The intuition behind this claim is that each recursive call diminishes the current distance to the target from  $\delta$  to at most  $3\delta/4$ . Therefore, after  $\log_{4/3} n = O(\log n)$  forwarding, the remaining distance between the current node

and the desired key will be reduced to at most  $2^m/n$ . The expected number of node landing in a range of this size is  $O(\log n)$  w.h.p. Thus, the remaining steps will reach the desired key within another  $O(\log n)$  steps. ■

## Experimental Results.

We ran a set of simulations to evaluate the average path length of Relaxed-2-Chord using the standard greedy routing. Our goal was to show that introducing a certain amount of heterogeneity in the network does not harm the routing performances.

During each experiment, a virtual ring is constructed by randomly generating the 20-bit IDs associated with the prescribed number of nodes. Then the routing table, for each node, is constructed by choosing for each finger a random point belonging to the corresponding interval. Finally, an all to all communication is simulated and accordingly the average path length is determined. The number of hops is counted for each route, and statistics are collected. 95% 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. Simulations done using up to  $2^{15}$  nodes, and identifier space of  $2^{20}$  IDs show that Relaxed-2-Chord has an average path length which is close to  $(\log n)/3$ .

This experimental result is not surprising since Manku et al., showed that, in the deterministic case (i.e., without randomness), the Chord protocol, having bidirectional links, provides an average path length equal to  $(\log n)/3 + \Theta(1)$  (cf. [10]). Therefore, the simulations have showed that the performances of our protocol do not get worse relaxing connections.

Furthermore, introducing a certain amount of randomization in the protocol allows us to exploit most performing routing strategies, like for instance the Neighbor of Neighbor (NoN) greedy routing (cf. [16]). Indeed, while it has been showed in [10] that the NoN approach is ineffective without randomization, on the other hand, the use of NoN

on an eclectic network allows to maintain the advantages of the greedy routing (cf. [22], for more details) while optimizing the tradeoff between routing table size and average path length.

We showed the asymptotical bound on the number of hops during a lookup operation because we want to be comparable with all the Chord-like systems, however our main goal is to provide a protocol which is able to reach optimal stretch. We address and prove this goal in the following. It is crucial for our strategy to make an initial amount of samples  $S$  in order to improve the stretch of the network. In [23] the authors proved the following Theorem which is directly applicable to our network.

**Theorem 2** [23] *If one follows a random sampling algorithm on an underlying graph  $G$  drawn from a family of graphs with  $d$ -power-law latency expansion, then the expected latency of a lookup is  $O(\Delta) + O(\Delta \log n/S^{1/d})$ .*

Therefore for  $S = \log^d n$  our strategy results in  $O(\Delta)$  expected latency, which is obviously the best we can do.

## 6. Conclusions

We have described Relaxed-2-Chord, a new DHT protocol, that is able to combine efficiency and scalability with a provable optimal stretch. The solutions, on this topic, that have come out during the last years can be divided in two sets; on one hand, we find those that use Landmark technique to measure the latency among nodes –this technique is difficult to manage in a distributed way and is not scalable; on the other hand, we have systems that try to construct overlays networks to reflect the underlying latency among nodes, unfortunately each node is subject to extra costs. The main contribution of our proposal is to build a low latency network using a parasitic strategy to sample nodes in the network without adding extra costs. The proposed network takes advantage from its symmetry. Each node maintains a routing table with up to  $2m - 2$  entries: clockwise and counterclockwise connections. The bidirectionality allows a node  $v$  connected to a node  $u$  to become a valid sample for node  $u$ , in particular, when  $u$  is the  $i^{th}$   $v$ 's forward (resp. back) finger then  $u$  can sample  $v$  and consequently may update its  $i^{th}$  back (resp. forward) finger. The randomness introduced with Relaxed-2-Chord guarantees the necessary flexibility in the construction of routing table that allows us to sample nodes in different IDs range. We support the design of this new protocol with a theoretical analysis both for the lookup performance that we prove to be resolved in  $O(\log n)$  hops w.h.p, and for the expected latency for lookup that has been showed to be  $O(\Delta) + O(\Delta \log n/S^{1/d})$  when the underlying network

follows a  $d$ -powerlaw latency expansion (which is, actually, the most expressive expansion to model a real Peer-to-peer network).

We leave as future work a more extensive evaluation Section, which will show in more details the gains of our system on the other Chord-like protocols and in particular the saving of our sample strategy compared to LPRS.

## References

- [1] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. “Practical Locality-Awareness for Large Scale Information Sharing”. In *Proc. of 4-th International Workshop on Peer-to-Peer Systems (IPTPS'05)* 2005.
- [2] I. Abraham, D. Malkhi, and O. Dobzinski. “LAND: Stretch  $(1 + \epsilon)$  Locality-Aware Networks for DHTs”. In *Proc. of 15-th Symposium on Discrete Algorithms (SODA'04)*, pages 550-559, 2004.
- [3] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, “Exploiting Network Proximity in Distributed Hash Tables”. In *Proc. of the Inter. Workshop on Future Directions in Distributed Computing (FuDiCo), Bertinoro, Italy*, Jun. 2002.
- [4] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, “Practical, distributed network coordinates”. In *Proc. of the 2nd Workshop on Hot Topics in Networks (HotNets-II), Cambridge, MA, ACM SIGCOMM*, Nov. 2003.
- [5] G. Cordasco, A. Negro, A. Sala, and V. Scarano, “PON: Exploiting Proximity on Overlay Networks”. In *Proc. of 4th International Workshop on Hot Topics in Peer-to-Peer Systems (HotP2P 2007)*, Mar. 2007.
- [6] G. Cordasco and A. Sala, “2-Chord Halved”. In *Proc. of 2nd International Workshop on Hot Topics in Peer-to-Peer Systems, (HOT-P2P 2005)*, pages 72–79, Jul. 2005.
- [7] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.
- [8] F. Dabek, J. Li, E. Sit, J. Robertson, F. Kaashoek, and R. Morris, “Designing a DHT for low latency and high throughput”. In *Proc. of the 1st Symposium on Networked System Design and Implementation (NSDI '04)*, Mar. 2004.
- [9] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker and I. Stoica, “The Impact of DHT Routing Geometry on Resilience and Proximity”. In *Proc. of ACM SIGCOMM*, Aug. 2003.

- [10] P. Ganesan and G. S. Manku, “Optimal Routing in Chord”. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, pages 176–185, Jan. 2004.
- [11] C. GauthierDickey, D. Zappala, V. Lo and J. Marr, “Low Latency and Cheat-proof Event Ordering for Peer-to-Peer Games”. In *Proc. of ACM NOSSDAV*, Jun. 2004.
- [12] D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewinand 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.
- [13] A. Kumar, S. Merugu, J. Xu, E. Zegura and X. Yu, “Ulysses: A Robust, Low-Diameter, Low-Latency Peer-to-Peer Network”. *European Transactions on Telecommunications on P2P Networking and P2P Services*, Vol. 15, No. 6, pp. 571-587, Dec. 2004.
- [14] L. Kleinrock, “Queueing Systems”. Vol. I and II, J. Wiley and Sons, 1975.
- [15] T. Locher, S. Schmid, and R. Wattenhofer, “eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System”. In *Proc. of 6-th IEEE Inter. Conference on Peer-to-Peer Computing (P2P'06)*, 2006.
- [16] G. S. Manku, M. Naor, and U. Wieder, “Know thy Neighbor’s Neighbor: The Power of Lookahead in Randomized P2P Networks”. In *Proc. of 36th thirty-sixth annual ACM Symposium on Theory of Computing (STOC '04)*, pages 54-63, Jun. 2004.
- [17] T. S. Eugene Ng and H. Zhang, “Predicting Internet network distance with coordinates-based approaches”. In *Proc. of IEEE Infocom*, 2002.
- [18] S. P. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-Addressable Network”. In *Proc. of ACM Special Interest Group on Data Communication (ACM SIGCOMM '01)*, pages 161–172, Aug. 2001.
- [19] S. P. Ratnasamy, “A Scalable Content-Addressable Network”. *PhD Thesis, University of California, Berkeley*, 2002.
- [20] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. 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, Feb. 2003.
- [21] E. Setton, J. Noh and B. Girod, “Low Latency video Streaming over Peer-to-Peer Networks”. In *Multimedia and Expo, 2006 IEEE International Conference on*, Jul. 2006.
- [22] J. Xu, A. Kumar, X. Yu, “On the Fundamental Trade-offs between Routing Table Size and Network Diameter in Peer-to-Peer Networks”. In *IEEE Journ. on Selected Areas in Comm., Volume 22, No. 1*, Jan. 2004.
- [23] H. Zhang, A. Goel, and R. Govindan, “Incrementally Improving Lookup Latency in distributed Hash Table Systems”. In *Proc. of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '03)*, 2003.
- [24] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: A Global-scale Overlay for Rapid Service Deployment”. In *IEEE Journal on Selected Areas in Communications (J-SAC)*, Vol. 22, No. 1, pages 41–53, Jan. 2004.