PON: Exploiting Proximity on Overlay Networks[*]

Gennaro Cordasco[†], Alberto Negro[†], Alessandra Sala[†] and Vittorio Scarano[†]

[†]ISISLab – Dipartimento di Informatica e Applicazioni "R.M.Capocelli"
Università degli studi di Salerno
via Ponte don Melillo - 84084 - Fisciano (SA), Italy.
E-mail: {cordasco,alberto,sala,vitsca}@dia.unisa.it

Abstract

We define a proximity overlay network (PON) which allow to realize DHT systems whose aim is to combine routing efficiency – i.e. an optimal degree/diameter tradeoff – and proximity awareness.

The proposed systems is parameterized with a positive integer s which measures the amount of flexibility offered by the network. Varying the value of s the system goes from a quite rigid network ($s = 2$) which offer an optimal degree/diameter tradeoff. Increasing s to relatively low values allows to increase the flexibility of the network and consequently improves the stretch, that is, the ratio between the latency of two nodes on the overlay network and the unicast latency between those nodes.

We are able to reconcile the conflict between the load balancing and proximity relationship by proving the efficiency of the main performance metrics. In particular we analytically prove that our system can result in lookup latencies proportional to the maximum latency of the underlying physical network, provided that the physical network has a power law latency expansion.

1 Introduction

Over the last six years, peer-to-peer (P2P) paradigm has generated a tremendous interest worldwide. Indeed, the phenomenon of file-sharing continues its sensational growth and seems to remain an important feature of the Internet for the immediate future. Furthermore, different instances of P2P architecture exist and offer services from Grid Computing to Distributed and Redundant File Storage, from Instant Messaging systems to Collaboration systems.

Traditional P2P networks, which exploit *consistent hashing* [8], are strongly based on an important abstraction: the homogeneity of nodes. Each node, in fact, is considered as being equally provided of computational capabilities, connection to other nodes, authentication required, bandwidth capability and so on. Furthermore for the sake of scalability, P2P systems [5, 14, 15, 20] are usually based on Distributed Hashing Tables (DHTs), which allows to identify and find resources in the system. DHTs naturally build up a homogeneous-structured overlay network, ignoring the heterogeneity nature of peers and connections.

On the other hand, in the "real world", nodes that join peer-to-peer networks are characterized by different properties, such as computing power and bandwidth as well as storage capacity. In this paper we propose a new peer-to-peer overlay network that, besides the traditional distributed protocols for lookup/join operation, is able to leverage on the nodes' heterogeneity in order to take advantage on the efficiency.

Our objective is to design a P2P overlay network which is able to work efficiently even in the (realistic) assumption that peers do not share the same capabilities, provided that they can be, somewhat, clustered so that the cost of a generic operation within a cluster (group) is smaller with respect the corresponding one outside groups. In this case, we say that the peers exhibit a *group property*. Several examples of group properties can be given, the first being, of course, the latency on the physical network that we will explicitly discuss in this paper: it is clear that communicating with a peer that is closer (in terms of latency) is less costly than communicating with distant peers. Bandwidth constraints could also represent a group property. This pa-

per, for sake of explanation, consider nodes' *proximity* in terms of their latency. Proximity represents, here, a generic cost function that defines how much should be charged for a communication — in the real physical network — between 2 nodes. Anyway, any similar property can be used.

Our work aims in designing a peer-to-peer network where nodes have a logical structure to the chosen property. In particular we exploit a landmarks technique [13] establishing some proximity property among peers. Peers with similar distances to some landmarks are considered close among them.

In general a DHT-based P2P system defines an Overlay Network, i.e., a virtual network of nodes and logical links, where each logical link correspond to multiple physical links. In this paper we design a Proximity Overlay Network (PON) which allows to keep logical links congruent to the underlying physical path, with no harm to the operational efficiency (e.g., load balancing, fault tolerance, scalability) of the considered Overlay Network.

Typically DHT schemes are based on consistent hashing [8] which allows to obtain several good properties, such as *load balancing*. Load balancing is critical to support high scalability, availability, accessibility, and throughput but, actually, each strategy that perfectly balance the load, looses on *proximity awareness*. As a matter of fact, consistent hashing is based on the randomization of the keys, so any proximity relation among peers is missed.

Our scheme is based on two different logical structure:

Topological ring: Each peer belongs to a ring built using a consistent hashing function, which allows to achieve the load balancing requirement;

Proximity space: Each peer lies on an auxiliary 1-dimensional space which preserves proximity relation among peers.

Thereafter, in order to speed up, the traditional lookup/join operations, some chordal links are added to the Topological ring accordingly with several information provided by the Proximity space.

We will measure the performance of our systems analyzing different metrics:

- the degree and diameter of the augmented topological ring. Indeed, each overlay network should have a small number of connections (small nodes' degree), so that only a limited amount of work need to be done for each alteration in the network. On the other hand, the system should provide a high connectivity, in order to be efficient and fault-tolerant. We will show that, while preserving proximity features, our scheme does not pay in terms of degree and diameter with respect several well known DHT-based schemes;

- the stretch of the system, which is the ratio between the latency of two nodes on the overlay path and the unicast latency between those nodes. We will show that the system uses the latencies of the underlying network to construct the topology so that the stretch will be optimal (i.e., constant.)

The remainder of this paper is organized as follows. Section 2 reviews some related work. Section 3 explains the idea at the basis of this paper. Section 4 reports a formal description of our proposal and presents theoretical results with respect to the degree/diameter tradeoff and the stretch. Section 5 shortly describes how to handle with nodes join, leave and fail. Finally, in Section 6 we conclude the paper with some final remarks.

2 Related Work

Earlier works [5, 14, 15, 20] have formed the main characteristics of DHT-based P2P networks: Peers have identifiers, taken from the same space as the keys (i.e., same number of digits). Each peer maintains a routing table consisting of a small subset of peers in the system. When a peer receives a query for a key t, the peer routes the query to its "closer" neighbor to t (i.e., the peer that makes the most progress towards resolving the query). The notion of *closer* differs from algorithm to algorithm, but in general, it depends on the distance metric function used in the system. These systems are also robust to frequent peers arrivals, departures and fails. Unfortunately, while these approaches guarantee a small number of overlay hops for lookup queries –generally polylogarithmic in the number of nodes in the systems– indeed, the network latency incurred for that queries can be quite high.

In order to reduce the stretch of that systems, some adjustments have been proposed. The key idea in [4] is to make the topology less rigid by introducing more than one choices for every link that is established. Unfortunately, the nodes that are examined to put in the routing tables represent only a small set of candidates, and therefore, it slowly converges to an optimal nodes' selection. In [13] the entire coordinate space is partitioned into equal size partitions, in such a way that peers close in terms of latency are assigned to the same partition. A consequence of the above technique is that the coordinate space is no longer uniformly distributed. 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 [10] builds a overlay that comprises both provable fault-tolerance and provable locality-awareness. However, this system lacks load balancing property when the underlying network results clustered in unbalanced manner; [16] and [18] propose

similar strategies for taking advantage of the inherent heterogeneity of the underlying physical network and consequently speed up routing. Unfortunately, these systems lack strategy to handle high churn rate.

The main contribution of our work is to combine the overall goal described above. Indeed, our system keeps the property of rigid networks: efficiency, load balancing, stability and fault-tolerance; moreover, it exhibits provable locality-awareness.

3 System overview

In a sense, our system resembles Chord [4] DHT systems, since PON is based on a ring of n nodes lying on a ring of 2^m identifiers – labeled from 0 to $2^m - 1$ in clockwise order. Node and key (resource) identifiers are obtained using consistent hashing [8]. Each node x has an m bit identifier (topological ID) and is connected with its predecessor $p(x)$ and its successor $s(x)$ on the ring. Consistent hashing assigns keys to nodes as follows. Key k is assigned to the first node whose identifier is equal to, or follows k on the ring.

Moreover, in order to maintain proximity information, each node lies over another auxiliary 1-dimensional space and, hence, has another m bit identifier (proximity ID) which reflects its physical position in the underlying network. The proximity ID is obtained using spacing filling curve [3] and a landmarks clustering technique [13]:

- A generic node x measures its latency to k well known landmark nodes that are randomly scattered in the underlying network;

- Let $\ell_i(x)$ the latency between the node x and the i-th landmark (for $i = 1, \ldots, k$). Position the node x into a k-dimensional space using $\langle \ell_1(x), \ell_2(x), \ldots, \ell_k(x) \rangle$ as its coordinate. (Observe that we could use this space as proximity space, but, unfortunately, the closeness among peers is not easy to compute on a high dimensional space);

- Use a standard algorithm to convert a higher dimensional space into a 1-dimensional one, which preserves the proximity relations among peers as shown in [21]. In particular we use a spacing filling curve [3] which allows to assign an m-bit proximity ID to each node in the network, in such a way that peers close in the underlying network are also close in terms of difference between their proximity ID.

Using the information maintained by the proximity space we are able to augment the topological ring in an efficient way, that is, we will design an overlay network (the augmented ring) which offer provable properties in terms of both degree, diameter and stretch.

Given an integer $0 < b < 2^m - 1$, we assume here that each peer x is able to identify/estimate the set (a.k.a. ball) $B_b(x)$ which contains the b peers closest to it. This assumption idealizes the hope that knowing the expansion of the network –by monitoring peers' past behaviors– can allow each node to estimate ball's diameter.

Let $b_0, b_1, \ldots, b_{r-1}$ an increasing set of integers (i.e. $0 < b_0 < b_1 < \ldots < b_{r-1} < 2^m - 1$) the topological ring is augmented picking, for each node x, a set of s nodes from each set $B_{b_i}(x)$ where i goes from 0 to $r - 1$.

Consequently each node x has a routing table with $r \times s$ entries (r rows/levels and s columns). By construction, the elements in the first levels provide small latencies and are used in the first steps of the routing process while the elements in the last levels could also connect peers that are far away in terms of latency and are usually used in the last routing steps (cfr. Figure 1).

We use a standard greedy routing strategy where, at each step, the routing query is forwarded to the neighbor which is the closer (non overshooting), in terms of distance on the topological ring, to the target key[1].
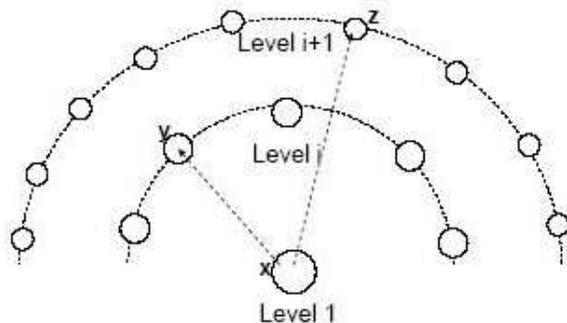

Figure 1. Each node x clusters the underlying network into r sets of concentric balls.

4 Routing scheme and performance metrics

The effectiveness of our proximity routing scheme depends on the size of the routing table used by each peer. Moreover, another important factor that influences the routing performance is the flexibility of the overlay network. Namely, algorithms – where routing table entries can be selected from a large set of candidates – are able to achieve

---

[1]The distance of a node, with topological ID x, to a target key t, with topological ID t is $(t - x) \bmod 2^m$ (the whole arithmetic on the ring is done $\bmod 2^m$.)

better performances, in terms of latency, than those that impose more constraints on routing table entries. On the other hand, flexible overlay networks are usually less efficient in terms of the degree/diameter tradeoff [7]. Our algorithm has a tuning parameter s which measures the overlay network's degree of flexibility.

As described in the Section 3 each node has two ID: A topological ID which is obtained using a consistent hashing function and guarantees a uniformly distribution of the nodes on the ring; A proximity ID which reflects the proximity relation among peers. Namely, peers close in terms of proximity ID are also close in terms of latency. Nevertheless, in order to improve system scalability, each node, according to several proximity information, builds up a routing table of up to $r \times s$ items.

The routing table of a generic node x is organized in r levels in such a way that connections to peers at level i are generally faster than connections to peers at level greater than i. Indeed, connections at level i consider only the b_i closest peers, while connections at a level greater than i could be slower since they consider a bigger ball around the node x.

Of course, the growing of the sequence $b_0, b_1, \ldots, b_{r-1}$ had to be done carefully; it does not make sense to increase disproportionately the size of one ball with respect the others. Moreover, the size of each ball must be big enough, in order to guarantee the presence of high number of candidate nodes (cfr. Lemma 1). In order to keep low the size of the routing table we choose a sequence $b_0, b_1, \ldots, b_{r-1}$ which grows exponentially and cover quite uniformly the whole ring ($[0, 2^m - 1]$). Therefore, in our system, the size of the i-th ball $B_{b_i}(x)$ (i.e. the candidates' set size to make a connection at level i) is

$$b_i = 2s^{i+1} \ln n.$$

Observe that the number of rows r is function of s. In particular the value of r is the minimum integer i such that the ball $B_{b_i}(x)$ cover all the nodes in the systems, that is $r = \min\{i \in \mathbb{N} \mid b_i > n\}$. Therefore, by choosing the value of s, we are fixing the routing table size ($r \times s$). Hence, the degree of our system (that is, the amount of memory used) depends only from the parameter s.

The smallest value of the parameter s is 2 which allow to design a quite-rigid overlay network that provides good performance with respect the degree-diameter tradeoff and a moderate improvements in terms of stretch. Indeed in this case we will show that both the degree and the diameter of the network are $O(\log n)$. Moreover, in order to obtain a more flexible network, we can also increase the value of s up to $\log n$. Bigger value of the parameter s provides a better improvements in terms of latency at the expense of a slightly bigger routing table.

Now we are ready to show the construction of the routing table for a node x. The first row of x's routing table keeps connection only with nodes lying in the smallest ball around x (i.e. nodes in $B_{b_0}(x)$). The whole network is partitioned into s intervals large $2^m/s$, such that w.h.p.[2] each of those s intervals contains at last one node $y_j \in B_{b_0}(x)$ for $j = 0, \ldots, s - 1$. Therefore the first row of x's routing table contains the peers y_j for $j = 0, \ldots, s - 1$.

Successively each subsequent row i is obtained as follows:

- Partition, the first interval considered at level $i-1$, into s subintervals large $2^m/s^{i+1}$;

- Pick a node, from each subinterval of size $2^m/s^{i+1}$, considering only nodes in $B_{b_i}(x)$ as candidate.

Obviously, this process is halted when the remaining interval does not contains any node. That is when $2^m/s^{r+1} < s(x) - x$. Since the expected distance between two successive node is $2^m/n$, we have that the expected number of row is $r = \log_s n$. Moreover, since the distance between two successive nodes is w.h.p. greater than $2^m/n^2$, we have that the number of row is, w.h.p., at most $2 \log_s n$. Consequently, in our scheme, each node holds a routing table of $O(s \log_s n)$ peers.

Formally, the routing table of a generic node x maintains, for each position (i, j), where $0 \le i \le r - 1$ and $0 \le j \le s - 1$, a connection to a node y where

$$y \in I_x(i, j) \stackrel{def}{=} \left[x + j\frac{2^m}{s^{i+1}}, x + (j+1)\frac{2^m}{s^{i+1}}\right[$$

and

$$y \in B_{b_i}(x).$$

Accordingly the level i of x's routing table covers the interval $\left[x, x + \frac{2^m}{s^i}\right[$ and connects with peers at a bounded latency (i.e. every connection at level i belongs to $B_{b_i}(x)$).

The following Lemma shows that the routing table outlined above is well defined, that is, for each item in the routing table of a generic node x, w.h.p. there is at least one candidate connection.

Lemma 1 *Let x be a generic node in the system, then for each range $I_x(i, j)$, w.h.p. there is at least a node y such that $y \in B_{b_i}(x)$.*

Proof. By definition the interval $I_x(i, j)$ has size $2^m/s^{i+1}$. Since the b_i nodes in $B_{b_i}(x)$ are scattered uniformly on the ring, we can analyze the problem as a set of b_i independent Bernoulli trials $X_1, X_2, \ldots X_{b_i}$, such that

$$X_i = \begin{cases} 1, \text{ if the } i\text{-th element of } B_{b_i}(x) \text{ belongs} \\ \quad \text{to } I_x(i, j) \\ 0, \text{ otherwise} \end{cases}$$

---

[2]Throughout this paper the term with high probability (w.h.p.) has been used to mean with probability at least $1 - \frac{c}{n}$ for some $c > 0$.

so let $p = Pr[X_i = 1] = \frac{|I_x(i,j)|}{2^m} = \frac{1}{s^{i+1}}$.

Let $X = \sum_{i=1}^{b_i} X_i$, we have that the expected number of nodes which belongs to B_{b_i} and lies in the interval $I_x(i,j)$ is $\mu = E[X] = E\left[\sum_{i=1}^{b_i} X_i\right] = \frac{b_i}{s^{i+1}} = \frac{2s^{i+1} \ln n}{s^{i+1}} = 2 \ln n$. Furthermore in the following we will show that with probability at least $1 - \frac{e}{n}$ there is at least one node in B_{b_i} which lies in $I_x(i,j)$.

According to *Chernoff Bound* [11] we know that:

$$Pr[X < (1 - \rho)\mu] < \left(e^{-\frac{\mu\rho^2}{2}}\right).$$

Hence, choosing $\rho = 1 - \frac{1}{\mu}$ we have:

$$
\begin{aligned}
Pr[X < 1] \quad &< \quad \left(e^{-\frac{\mu\left(1 - \frac{1}{\mu}\right)^2}{2}}\right) \\
&= \quad \left(e^{-\frac{(\mu-1)^2}{2\mu}}\right) \\
&< \quad e^{-(\mu/2)+1} = \frac{e}{n}.
\end{aligned}
$$

The following Theorem shows that the diameter of our schemes is $2r$.

Lemma 2 *For any value of $s > 2$, the diameter of our scheme with $n = 2^m$ nodes is at most $2r$.*

Proof. We can prove, by induction, that after performing $2i$ routing hops, the search interval is reduced to $2^m/s^i$. In fact if $i = 0$ the search interval is bounded by 2^m. Consider the situation after $2(k - 1)$ routing steps and let x the current node. By the inductive hypothesis we have that the search interval is at most $2^m/s^{k-1}$. Hence, since the level $(k - 1)$-th level of x's routing table cover the interval $[x, x + 2^m/s^{k-1}[$, using the greedy strategy, and in particular choosing a connection from the $(k - 1)$-th level of x's routing table, in one hop, we are able to restrict the search interval[3] to $2^{m+1}/s^k - 1$. Thereafter, using one more connection (from level $k-1$ or k) – that is, after $2k$ routing hops – the search interval is reduced to $2^m/s^k$. For $i = r$, i.e., after at most $2r$ routing steps, the remaining interval is thus reduced to $2^m/s^r$. Hence by observing that $r = log_s 2^m$, we have $2^m/s^r = 1$ and the search ends. $\square$

We can also generalize the results to hold in a ring where not all nodes are present. Due to consistent hashing constraints [9] the n nodes can be assumed to be uniformly distributed. Therefore we can state the following.

---

[3]Observe that the distance between two consecutive nodes which belongs to a generic level i of a routing table is at most $2 \times 2^m/s^{i+1} - 1$.

Theorem 1 *For any value of $2 < s \leq \log n$, the diameter of our scheme with $n < 2^m$ nodes is $O(r)$.*

Proof. Consider a source that wants to send a message at distance d. From Lemma 2 it follows that diminishing the distance to size $2^m/n$ takes $O(r)$ hops. Left is to prove that the number of node alive in an interval I of size $2^m/n$ is small. The expected number of nodes alive in the interval is 1. Furthermore, with probability larger than $1 - 1/n^2$, the number of nodes that lies in the same interval is $O(\log n/\log \log n)$, see example 4.4 in [11]. Therefore the maximum number of routing steps is $O(r + \log n/\log \log n) = O(r)$ when $s \leq \log n$. $\square$

Using the same argument of Lemma 2 it is easy to show that the average path length provided by our scheme is approximatively $\frac{s-1}{s} \log_s n$.

The last step to complete the argumentation for the routing performance is to show the value of the stretch. Let x_1 be a lookup starting point, and let x_k be the target traversed in the lookup route. Then the stretch is $\frac{l(x_1,x_2)+...+l(x_{k-1},x_k)}{l(x_1,x_k)}$ where $l(x,y)$ is a cost function that measures the unicast latency between x and y.

Before proceeding, we need to introduce the concept of d-powerlaw latency expansion: Let $N_\delta(x)$ denote the number of nodes in the network that are within latency δ of x. Informally, for $d \geq 1$, a family of graphs has a d-powerlaw latency expansion if $N_\delta(x)$ grows (i.e. "expands") proportionally to δ^d, for all nodes x. Some simple examples of graph families with power-law latency expansion are rings ($d = 1$), lines ($d = 1$), and meshes ($d = 2$).

Theorem 2 *Assuming that our scheme is drawn from a family of graphs with a d-powerlaw latency expansion, the expected stretch of our scheme, with $s \geq 2^d$, is smaller than 4.*

Proof. Let Δ be the maximum physical connection cost between each couple of nodes in the network. According to our strategy each node clusters the physical network into r balls. Afterwards the node is able to set up its routing table, particularly each routing table level is built considering only nodes in the corresponding ball. By Lemma 1 we know that, in the worst case, each lookup is composed of at most 2 hops for each routing table level. Since for each $i = 1, \ldots, r - 1$ we have that $b_{i-1} = b_i/s$ and the considered underlying network follows a d-powerlaw latency expansion, we have that the the latency between a generic node and a node in the i-th level of its routing table is at most $\frac{\Delta}{s^{(r-i-1)/d}}$ which is smaller than $\frac{\Delta}{2^{(r-i-1)}}$ when $s \geq 2^d$.

Thus, the maximum latency for a routing path is smaller than

$$2\sum_{i=0}^{r-1} \frac{\Delta}{2^{(r-i-1)}} = 2\sum_{i=0}^{r-1} \frac{\Delta}{2^i} < 4\Delta$$

s	Average Routing Table Size	$n \approx 2^{20}$	Diameter	$n \approx 2^{20}$	Bounded Stretch (cfr. Th. 2) (d-power-law expansion)	$n \approx 2^{20}$
2	$2 \log n$	≈ 40	$2 \log n$	≈ 40	$d \leq 1$	$d \leq 1$
$\log \log n$	$\frac{\log n \log \log n}{\log \log \log n}$	≈ 41	$\frac{2 \log n}{\log \log \log n}$	≈ 19	$d \leq \log \log \log n$	$d \leq 2.11$
$\log n$	$\frac{\log^2 n}{\log \log n}$	≈ 92	$2 \frac{\log n}{\log \log n}$	≈ 9	$d \leq \log \log n$	$d \leq 4.3$

Table 1. Performances of PON with respect to degree, diameter and stretch.

Consequently the value of the stretch is: $\frac{l(x_1,x_2)+l(x_2,x_3)+...+l(x_{2r-1},x_{2r})}{l(x_1,x_{2r})} < \frac{4\Delta}{\Delta}$ and so the stretch is smaller than 4.

$\square$

5 Node bootstrap and churn environment

The last step to complete the description of our work is to show that our protocol is stable and efficient even in a high churn rate environment. Theoretical analysis proves that our system works well assuming the correctness of routing table (cfr. Section 4). Moreover, using the same arguments showed in [4], is possible to prove that the routing performances degrade gracefully even if a significant amount of the routing table entry is incorrect.

When a node join the network, it needs to set up its routing table. Our systems exploits a fast bootstrap strategy: the new node asks several neighbor about their topological and proximity ID, in order to select a node that is quite close to it with respect to both its ID. Then it uses the routing table of the selected node as a starting point to build an updated routing table. Afterwards, the routing table is kept up to date by piggybacking on lookup operation. As shown in [19] this strategy converges quickly to an optimal configuration of the routing table entries.

6 Conclusions

We have described a Proximity Overlay Network design technique for DHT systems based on geometric routing. Our protocol shows as main contribution the capability to combine load-balancing, locality-awareness and fault-tolerance: our system is based on an overlay network, i.e., a virtual network of nodes and logical links, built using consistent hashing, in order to guarantees several properties such as load-balancing and fault–tolerance; on the other hand, our system exploits a proximity space to keep logical links congruent to the underlying physical path, and consequently, shows improved lookup latencies.

Our scheme provides a tuning parameter s which determines the degree of flexibility of the overlay network in the sense of [7]. Varying the value of s the system goes from a quite rigid network ($s = 2$) which offer an optimal degree/diameter tradeoff. Increasing s to relatively low values allows to increase the flexibility of the network and consequently improves the stretch. We have analytically shown that on graphs with d-power-law latency expansion, our system – with parameter $s \geq 2^d$ – can result in an average lookup latency that is proportional to the maximum unicast latency. Table 1 shows the performances of our system, considering three values of the parameter s. Moreover, in order to give an idea of the system's scalability, we report also the performances that could be obtained in large networks. Indeed, each table entry presents also the value of the corresponding metric when the number of nodes alive in the network is approximatively 2^{20}.

References

[1] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. "Practical Locality-Awareness for Large Scale Information Sharing". In *Proceedings 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 *Proceedings of* 15-*th Symposium on Discrete Algorithms (SODA'04), pages 550-559*, 2004.

[3] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmaier. "Space Filling Curves and Their Use in the Design of Geometric Data Structures". In *Theoretical Computer Science, vol. 181, num. 1, pages 3-15*, 1997.

[4] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. "Exploiting Network Proximity in Distributed Hash Tables". In *Proceedings of the International Workshop on Future Directions in Distributed Computing (FuDiCo), Bertinoro, Italy*, 2002.

[5] P. Druschel, and A. Rowstron. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In *Proceedings of the* 18-*th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, pages 329–350*, 2001.

[6] P. B. Godfrey, and I. Stoica. "Heterogeneity and Load Balance in Distributed Hash Tables". In *Proceedings of of the* 24-*th International Conference on Computer Communications (INFOCOM'05)*, 2005.

[7] K. Gummadi and R. Gummadi and S. Gribble and S. P. Ratnasamy and S. Shenker and I. Stoica. "The impact of DHT routing geometry on resilience and proximity". In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM '03*, 2003

[8] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. 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* 29-*th Annual ACM Symposium on Theory of Computing, ACM Press, El Paso, TX, USA, pages 654–663*, 1997.

[9] Daniel 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/.

[10] T. Locher, S. Schmid, and R. Wattenhofer, "eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System". In *Proceedings of* 6-*th IEEE International Conference on Peer-to-Peer Computing (P2P'06), Cambridge, United Kingdom*, 2006.

[11] R. Motwani, and P. Raghavan. "*Randomized Algorithms*". Cambridge University Press, 1995.

[12] S. P. Ratnasamy. "A Scalable Content-Addressable Network". Ph.D. Thesis, University of California at Berkeley, 2002. http://berkeley.intel-research.net/sylvia/thesis.pdf.

[13] S. P. Ratnasamy, M. Handley, R. Karp, and S. Shenker. "Topologically-Aware Overlay Construction and Server Selection". In *Proceedings of of the* 21-*th International Conference on Computer Communications (INFOCOM'02), New York, NY*, 2002.

[14] S. P. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A scalable content-addressable network". In *Proceedings of ACM Special Interest Group on Data Communication (ACM SIGCOMM '01), San Diego, CA, US, pages 161–172*, 2001.

[15] 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*, 2003.

[16] Z. Xu, C. Tang, and Z. Zhang. "Building Topology-Aware Overlays Using Global Soft-State". HP Labs, Tech. Rep. HPL-2002-281, 2002.

[17] Z. Xu and Z. Zhang. "Building Low-maintenance Expressways for P2P Systems". Hewlett-Packard Labs: Palo Alto, 2001.

[18] Z. Xu, M. Mahalingam, and M. Karlsson. "Turning Heterogeneity to an Advantage in Overlay Routing". HPL- 2002-126R1, 2002.

[19] H. Zhang, A. Goal, and R. Govindan. "Incrementally improving lookup latency in distributed hash table systems". In ACM Sigmetrics, 2003.

[20] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An infrastructure for fault-tolerant wide-area location and routing". In *Tech. Report No. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley, California 94720, USA*, 2001.

[21] Y. Zhu, and Y. Hu. "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems". In *IEEE Transaction on Parallel and Distributed Systems, vol. 16, no. 4*, 2005.