# LAND: Stretch $(1+\varepsilon)$ Locality-Aware Networks for DHTs

Ittai Abraham, Dahlia Malkhi*and Oren Dobzinski
The Hebrew University of Jerusalem, Israel
{ittaia,dalia,orend}@cs.huji.ac.il

**Abstract**

This paper proposes the first peer-to-peer network and lookup algorithm that for any $0 < \varepsilon$ has worst case stretch bounded by $1+\varepsilon$. The construction uses an expected logarithmic number of links. It is suitable for a very realistic class of metrics in which the only restriction on density is a growth-bound. It is completely decentralized and readily deployable in dynamic networks.

## 1 Introduction

One of the main challenges of distributed systems is how to efficiently store and locate ever increasing amounts of content. The tremendous growth of the Internet illuminates this challenge. In the traditional basic web architecture, all requests reach the same server. This single service-point architecture may result in network congestion or server swamping as the number of requests increases.

The solution suggested by Karger et al. [9] is to use consistent hashing. Using a hash function, web page URLs are hashed, and from the hashed value a close-by cache server containing the page is accessed. Instead of having just one cache server (and suffer from scalability and fault tolerance problems), multiple caching servers act together to form a Distributed Hash Table (DHT). In the DHT paradigm, a lookup request is routed through the server network to the specific server that knows the answer to the lookup query.

DHTs are also at the focus of rising interest in file-sharing programs like Gnutella, Napster, and Freenet. These motivated a growing interest (e.g., see [19, 4, 22, 18, 11, 14, 13, 12, 1, 16, 15, 5, 3, 8]) in the field of distributed computing to the challenges of building DHTs: scale, dynamism, fault tolerance, decentralized control.

**A case for locality-awareness.** In many of the DHT works, one of the main measures of quality is the number of hops taken until the server with the required information is found. In most cases, logarithmic complexity (e.g., in [19, 4, 22, 14]) or polylogarithmic complexity (e.g., in [15]) is sought. If the only bound on reaching the target is that the number of hops is bounded, say by 4, then the lookup request could go, for example, from Boston through New Zealand, Brazil, France and finally to New York. Although the number of hops is small in this example, this clearly is not a desired outcome. While bounding the number of hops by a logarithm is important, several works [17, 18, 4, 22, 11, 3] have argued that a far more important measure is the total cost of communication between peers.

The natural way to model costs is to assume a cost function $c$ that induces a metric space on the universe of servers. The main measure of a routing protocol is it's *stretch*, namely, the ratio between the distance and the cost of a route. More precisely, let $x$ be a lookup starting point, $y$ be the target of the lookup (i.e., the closest node containing the searched object). Let $x = x_1, x_2, \ldots, x_k = y$ be the nodes traversed in the lookup route. Then the stretch is $\frac{c(x_1,x_2)+\ldots+c(x_{k-1},x_k)}{c(x,y)}$. The seminal work of Plaxton, Rajaraman, and Richa [17] is one of the first works to provide a distributed lookup protocol with analytical bounds on the stretch. They present a randomized scheme for a class of metric spaces representing realistic networks, in which the expected stretch in finding targets is a (rather large) constant. Several efforts were made to deploy this scheme, e.g., Tapestry [22] and Pastry [4]. These systems construct a dynamic DHT based on the principles of [17], yielding efficient, locality-aware lookup programs. Recently, Li and Plaxton introduce in [11] a simplified version of the PRR scheme, that yields lookup costs proportional to the diameter of the network (rather than proportional to the distance between $x$ and $y$).

Our work builds on the successful approach introduced and deployed in [17, 22, 4, 18, 11]. It enhances it by presenting the first constant $1+\varepsilon$ stretch DHT, in which for all routes the cost ratio between the distance and the route is guaranteed to be at most $1+\varepsilon$. The guarantee of constant stretch is achieved while not

*This work was supported in part by the Israeli Science Foundation (246/02) and by EC Evergrow project.

impairing on other parameters of the network such as degree, memory requirements, adaptability, and fault tolerance.

Locality aware DHT can be viewed as distributed load balanced designs of compact routing schemes [21], and are related to distance oracles [20]. For general metrics, compact routing schemes have stretch of at least 3 [6]. In contrast, for growth bounded metrics[1] we show that stretch $1+\varepsilon$ is achievable in a distributed and load balanced scheme using only an expected logarithmic amount of memory per node [2].

**Overview of the scheme.** As in several DHTs, nodes have virtual identifiers composed of string-identifiers over some base $B$. Objects also have virtual identifiers. Routing to a virtual address is done via *prefix routing*, namely, the $i$'th routing step fixes the $i$'th digit of the address. Likewise, objects publish their existence via prefix routing destined at their virtual address. Similar to [17], routing steps geometrically increase in cost as we fix more digits. Intuitively, this stems from the fact that there are expected $n/B^i$ nodes matching $i$ prefix digits in a network with $n$ nodes. Thus, the more digits are fixed, the harder it is to find a suitable node to hop to. The total distance taken by $i$ routing steps forms a geometric series that is dominated by the $i$'th step.

The novelty of our scheme lies in the choice of links, and in the analysis of the routing algorithm. The links of the network are chosen so that each node "hosts" $\log n$ *routing entities (routers)*, each one allowing it to "fix" one of its identifier digits to any desired value. Unlike previous schemes, our choice of links **enforces** a distance upper bound on each stage of the route, rather than probabilistically maintaining it. This is done by having routers choose their links within an appropriately bounded distance. If no suitable endpoint is found for a particular link, we suggest a novel technique in which the node *shadows* a router responsible for routing onward one more digit. The splitting of a node into distinct routing entities, each one responsible for only one digit position, is crucial for the emulation to work properly. It allows bounding the amount of links emulated by each node to an expected logarithm.

Additionally, a node to which an object is published puts references to the object's location within its vicinity. Specifically, the node reached in the $i$'th step of publishing stores a reference on all nodes matching the first $i$ digits of its identifier within a certain distance. The purpose of these reference copies is to terminate the route quickly when the route from $s$ to $t$ already goes a distance roughly $\varepsilon \cdot c(s,t)$, in order to keep the stretch bounded by a constant. Here again, our scheme differs from previous methods in that the references are guaranteed to be maintained to within a certain distance, not probabilistically so.

Routing is done in two stages. Starting from a node, the first stage performs *prefix-routing*, fixing one digit after another in the address of the target. It terminates when a node containing a reference to the object is reached. The second stage consists of simply following the reference pointers until the target is found.

Our analysis shows that the cost of a route from $s$ to $t$ is deterministically bounded by $(1+\varepsilon)c(s,t)$. Similar to [10], our locality analysis is given in a realistic network model whose density is *growth bounded* from above. (The network model of [17, 4, 22] is somewhat more restricted and assumes a density bound from below as well.)

**Contribution.** This paper presents a novel DHT construction that maintains locality in lookup operations. We borrow heavily from the principles of [17], yet our design has the following important achievements:

- Most importantly, our stretch factor can be chosen to be arbitrary low on all lookup routes. We deterministically bound the cost of lookups in our scheme to $1+\varepsilon$ for any $0 < \varepsilon$, while the cost bounding analysis of [17] archives only a large constant in *expectation*.
- The scheme in [17] assumes, in addition to a growth bound, also a lower bound on the expansion, a shrink bound[3]. Our construction and analysis eliminates the need for a shrink bound and works on the larger class of metric spaces that only have the property of being growth bounded.
- Our construction is simple, and our proofs short and intuitive. In contrast, proving the expected locality of [17] requires an involved and lengthy analysis. In our belief, the simplicity of our scheme and the elegance of its analysis may lead to improved practical deployments.
- We present an additional construction in Section 6 that embraces the two-tier architecture of current peer-to-peer networks, where stronger and stable nodes serve as ultra-peers, and other (e.g., transient home users) are regular peers. Under this model,

[1] Growth bounded metrics: for any point $x$ and distance $r \geq 1$ the number of points within distance $2d$ of $x$ is at most a constant factor larger than the number of points within distance $r$.

[2] Derandomization of this scheme will appear in the full paper.

[3] Shrink bounded metrics: for any point $x$ and distance $r \geq 1$ the number of points within distance $2d$ of $x$ is at least a constant factor larger than the number of points within distance $r$.

we improve our scheme parameters to employ only an expected constant out-degree for nodes. The construction yields a guaranteed stretch bounded by $2+\varepsilon$. Our two-tier construction is the first constant-degree locality-aware scheme, compared with all previous locality-aware schemes that have logarithmic degree.

## 2 Preliminaries

The system consists of a set of nodes denoted $V$, where the number of all nodes $|V| = n$. Let $c : V^2 \mapsto \mathbb{R}_+$ be the cost function, associated with pairs of nodes, that expresses the cost of communication between nodes. We assume $c$ has positivity, reflexivity, triangle inequality, and symmetry. Thus $< V, c >$ forms a metric space. From here on, we refer to the cost as the *distance* between nodes.

**Minimum density** We assume that the minimal distance between every pair of nodes is 1.

**Growth bounded metrics** are considered, e.g., in [10] and reflect real-life internets. The set of nodes within distance $r$ from $x$ is denoted $N(x,r)$. For *growth bounded metrics*, we assume an expansion constant $\Delta$, such that for every node $x$ and $r \geq 1$, we have

$$|N(x,2r)| \leq \Delta |N(x,r)| \ . \tag{2.1}$$

We note that growth bounded metrics are more general than the metrics assumed in [17, 4, 22], which also bound from below the shrinking rate of increasing neighborhoods.

**Nodes and routers** Each node $u$ hosts an assembly of *routing entities (routers)*. A router $r$ hosted by node $u$ is denoted $u.r$. Each router is identified by a string $r.id$. When clear from the context, we sometimes refer to $u.r$ simply as $r$. Identifier strings are composed of $M$ digits in radix $B = 2^b$. The radix $B$ is chosen such that $B \geq \Delta^2$. For a network with $n$ nodes the length of each identifier, $M$, is chosen such that $M = \lceil x \rceil$ for $x$ satisfying $B^x = n$. A router $u.r$ has an additional *level* property, denoted $u.r.level$, between 1 and $M+1$.

**Notations for bit sequences** Let $s$ be a $k$-digit identifier. Denote $s[j]$ as the prefix of the $j$ most-significant digits, and denote $s_j$ as the $j$'th digit of $s$. A concatenation of two strings $s, s'$ is denoted by $s||s'$.

**Summary of notations and constants** All notations and definitions used in our construction (including those mentioned above already) are summarized for convenience here.

- Radix is at least square of expansion $B \geq \Delta^2$.
- Number of identifier-digits $M = \lceil x \rceil$ for $x$ s.t. $B^x = n$.
- The constant $\alpha$ is chosen such that $Be^{-\alpha} < 1$.
- Let $A_i(x)$ denote the smallest ball around $x$ containing $\min(\alpha B^i, n)$ nodes, let $a_i(x)$ denote the radius of $A_i(x)$.
- Denote $\gamma = B^{\log_\Delta 2}$ and note that $\gamma \geq 4$.
- For $\varepsilon$ the desired stretch, choose $d$ as in Equation 5.2.

**Useful properties** The main properties implied by the growth-bound assumption are summarized in the following technical lemma. These properties suffice by themselves to uphold the LAND construction, and from here on we refer to the network properties only through them.

LEMMA 2.1. *Let $x$ and $y$ be any two nodes, for any $i$ such that $y \in A_i(x)$:*
*(i) $A_i(x) \subseteq A_{i+1}(y)$.*
*(ii) $A_i(y) \subseteq A_{i+1}(x)$.*
*(iii) $a_{i+1}(x) \geq \gamma\ a_i(x)$, where $\gamma = B^{\log_\Delta 2}$.*

*Proof.* Let $r = a_i(x)$ denote the radius of $A_i(x)$, so $A_i(x) = N(x,r)$, where $N(x,r) = \{v \in V \mid c(x,v) \leq r\}$. Since $y \in N(x,r)$ then (see Figure 1 )

$$N(x,r) \subseteq N(y,2r) \subseteq N(x,3r) \ .$$

From the growth bounded assumption we can bound the number of nodes in $N(x,3r)$ using $|N(x,r)|$ as follows: $|N(x,3r)| \leq \Delta^2 |N(x,r)| = \Delta^2 |A_i(x)| = \Delta^2 \alpha B^i \leq \alpha B^{i+1}$ .

For (i), $N(x,3r) \subseteq A_{i+1}(x)$, and so $A_{i+1}(y)$, the ball around $y$ with $\alpha B^{i+1}$ nodes, must contain $N(y,2r)$ and so must contain $A_i(x)$. For (ii), $A_i(y) \subseteq N(y,2r) \subseteq N(x,3r) \subseteq A_{i+1}(x)$.

For (iii), $B^{\log_\delta 2} = 2^{log_\Delta B}$, hence $|N(x, B^{\log_\Delta 2} r)| \leq \Delta^{\log_\Delta B} |N(x,r)| = \alpha B^{i+1}$, so $A_{i+1}(x) \supseteq N(x, B^{\log_\Delta 2} r)$. □

## 3 LAND Architecture

The goal of our work is to support a lookup operation that locates nearest copies of objects, such that the nodes of the network share the lookup load evenly. Our method adopts the distributed hash table (DHT) approach that is employed in several recent systems [19, 4, 22, 14].

More formally, let $\mathcal{A}$ be a set of objects. We make use of a uniformly distributed hash function $H$ on object names. For any object $obj \in \mathcal{A}$, such that $obj$ is stored on some node $s$, reference information about $obj$'s location is stored on nodes whose identifiers match prefixes of various length of $H(obj)$. The object $obj$ may be replicated in various locations in the network. Our network is able to locate a nearby copy of every $obj \in \mathcal{A}$.

All nodes in the network take part in the routing algorithm, and pass queries to the nodes they have outgoing links to. All nodes store partial reference information about the location of objects. In this sense, the nodes form a *Distributed Hash Table*.

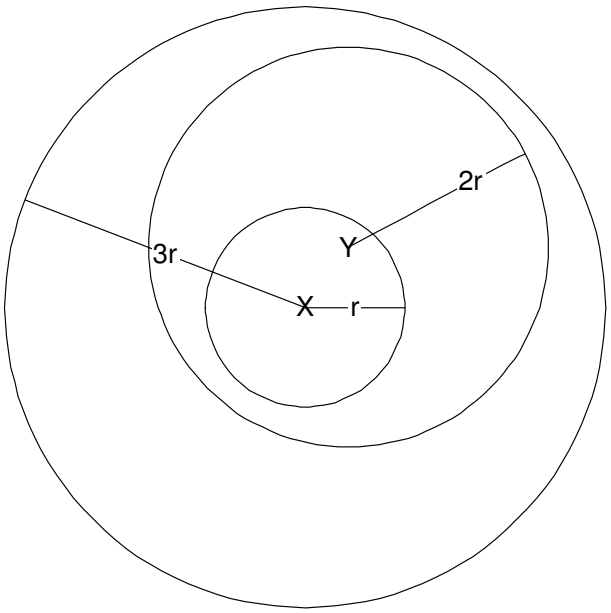


Figure 1: The circles $N(x,r)$, $N(y,2r)$, and $N(x,3r)$

More precisely, the basic entity in our network is the *router entity* (simply referred to as 'router'). Each router has an identifier and level. The routers are connected in a manner similar to the butterfly graph, i.e., level $\ell$ routers have outgoing links only to level $\ell+1$ routers. In LAND each node in the system initially maintains $M+1$ routers, each with a different level from 1 to $M+1$. In addition, it is possible that some nodes will need to maintain additional *shadow routers* to ensure worst case locality.

When we mention a router $r$, we interchangeably mean either the routing entity itself or the node that hosts the router. The precise meaning will be evident from the context.

## 4 The Routing Network

Each node $v$ hosts an initial set of $M+1$ routers denoted $R_1, ..., R_{M+1}$. Router $v.R_i$ has identity $v.R_i.id$ and level $i$, i.e., $R_i.level = i$. The identifiers of routers are represented as radix $B = 2^b$ numbers. The identifiers of the initial routers are randomly chosen uniformly and independently.

Recall that the constant $B$ is chosen such that $B \geq \Delta^2$. In addition a constant $\alpha$ is chosen such that $Be^{-\alpha} < 1$. We denote the number of identifier digits as $M$ chosen such that $B^M = n$. Recall that $A_i(x)$ denotes the smallest ball around $x$ containing $\min(\alpha B^i, n)$ nodes. Recall that $a_i(x)$ denotes the radius of $A_i(x)$.

Let $r$ be a router of level $r.level = \ell$ hosted by node $v$. The router $r$ could be either the initial $v.R_\ell$, or a shadow router hosted by $v$, as we shall see below. Router $r$ has outgoing links of two types, *neighbor* and *publish* denoted $r.L$ and $r.P$ respectively. The outgoing links of $r$ are as follows:

**neighbor:** If $\ell \leq M$, then router $r$ has $B$ neighbor links, denoted $L(0), ..., L(B-1)$. The $i$'th neighbor $r.L(i)$ is selected as the closest router within $C_i(r) \cap A_\ell(v)$, where $C_i(r) = \{u \in V \mid \exists s, u.s.id[\ell] = r.id[\ell-1]||i, u.s.level = \ell+1\}$. The link $L(i)$ 'fixes' the $\ell$'th digit of $r.id$ to $i$, namely, it connects to the closest node $u$ that hosts a level $\ell+1$ router $u.s$ that matches the id $r.id[\ell-1]||i$, within the ball $A_\ell(v)$ (i.e., among the $\alpha B^\ell$ closest nodes to $v$).

If $C_i(r) \cap A_\ell(v) = \emptyset$ then node $v$ hosts a *shadow router* $s$ with identifier $r.id[\ell-1]||i$ and level $\ell+1$. Node $v$ maintains all of the links of the shadow router (including the *publish* links described below).

Since a shadow router also requires its own neighbor links, it may be that the $j$th neighbor link of a shadow router $s$ does not exist in $C_j(s) \cap A_{s.level}(v)$. In such a case $v$ also hosts a shadow router that acts as the $s.L(j)$ endpoint.

Shadow hosting continues recursively until all links of all the shadow routers hosted by $v$ are found (or until the limit of $M+1$ levels is reached).

**publish:** If $\ell \leq M$, the publish links $r.P$ are all the nodes hosting any level-$(\ell+1)$ router with the same first $\ell-1$ bits as $r.id$ which are inside the ball $A_{\ell+d+5}(v)$. Formally, $r.P = C(r) \cap A_{\ell+d+5}(v)$, where $C(r) = \{u \in V \mid \exists s, u.s.id[\ell-1] = r[\ell-1], u.s.level = \ell+1\}$.

**4.1 Publish and lookup** The publishing of an object $obj$ residing on a node $t$ proceeds as follows. Starting with a level-1 router $w_1.r$ (where $w_1 = t$, move from a node $w_i$ using the neighbor links of the level $i$ router $w_i.r$ by fixing the $i$'th digit to that of $H(obj)$. This links to $w_{i+1}$, a node hosting a level $i+1$ router $w_{i+1}.r$ (this might be a shadow router in which case $w_i = w_{i+1}$) such that $w_{i+1}.r.id[i] = H(obj)[i]$. Thus, router $w_{i+1}.r$ has level $w_{i+1}.r.level = i+1$, and id $w_{i+1}.r.id[i] = H(obj)[i]$. Continue until the $M$'th digit (i.e., until there are no more neighbor links to follow).

Each node $w_i$ along the publishing route stores a reference to $obj$ which points back to $w_{i-1}$. In addition, $w_i$ stores such a reference on every node of $w_i.r.P$.

A lookup operation of an object $obj \in \mathcal{A}$ can be initiated by any node in the system, and its purpose is to find the closest node storing $obj$. The lookup

operation from a node $x$ proceeds in two stages. The first stage fixes target digits one by one. The loop goes as follows: Starting with a level-1 router at $x$ denoted $x_1.r$, and so long as the target was not found, then from the current router $x_i.r$, first check if there is a reference to *obj* with a link to $w_{i-1}$. If so, move to $w_{i-1}$ and continue with the second phase. Otherwise, if $H(obj)_i$ is $j$, continue at a node $w_{i+1}$ with a router $w_{i+1}.r$ such that $w_{i+1}.r.level = i+1$ and $w_{i+1}.r.id[i] = H(obj)[i]$ (this might be a shadow router).

The second stage traverses from $w_{i-1}$ backward to $t$ using *obj*'s reference links.

The publish and lookup algorithms for a router $u$ are provided in pseudo-code in Figure 2 below.

---

A node $t$ that wants to store an object *obj* initiates $t.R_1.\text{PUBLISH}(obj, t, 1)$.

PUBLISH $(obj, w, \ell)$ at router $u.r$:
  store "*obj*;$w$" on node $u$;
  send "*obj*;$u$" to every node in $u.r.P$;
  if $\ell \leq M$
    $u.r.L(H(obj)_\ell).\text{PUBLISH}(obj, u, \ell+1)$;

---

A node $x$ that wants to lookup object *obj* initiates $x.R_1.\text{LOOKUP}(obj, x, 1)$.

LOOKUP $(obj, x, \ell)$ at router $u.r$:
  if $u$ stores *obj*
    return *obj* to $x$;
  else if $u$ stores "*obj*;$v$"
    $v.\text{LOOKUP}(obj, x, \ell)$;
  else if $\ell \leq M$
    $u.r.L(H(obj)_\ell).\text{LOOKUP}(obj, x, \ell+1)$;

---

Figure 2: The PUBLISH and LOOKUP algorithms.

## 5 Analysis

### 5.1 Expected degree

LEMMA 5.1. *For every initial router $R_\ell$ hosted by a node $v$ the expected number of shadow routers hosted by $v$ due to $R_\ell$ is constant.*

*Proof.* For any level $1 \leq k \leq M$, the probability that link $L(i)$ will be found inside $A_k(u)$ is at least $1 - \left(1 - \frac{1}{B^k}\right)^{\alpha B^k} \geq 1 - e^{-\alpha}$.

For $0 \leq i \leq M - \ell$, let $b_{\ell+i}$ be a random variable that counts the number of level-$(\ell+i)$ shadow routers that $u$ recursively emulates due to missing links. Such shadow routers are created if $u$ incurs emulation of a level-$(\ell+1)$ shadow router; one of that shadow router's links is also emulated by a level-$(\ell+2)$ shadow router; and so on, up to level $(\ell+i)$.

So $b_\ell = 1$ and for $1 \leq i \leq M - \ell$, each of the $b_{\ell+i-1}$ routers has $B$ neighbor links with a probability of emulating each one bounded by $e^{-\alpha}$. Therefore $E[b_{\ell+i} \mid b_{\ell+i-1}] \leq b_{\ell+i-1}Be^{-\alpha}$ and due to the independence of the identifiers $E[b_{\ell+i}] \leq E[b_{\ell+i-1}]Be^{-\alpha}$. Thus by induction $E[b_{\ell+i}] \leq (Be^{-\alpha})^i$.

The expected total number of shadow routers incurred by router $u$ is bounded by:

$$E[\sum_{0 \leq i \leq M-\ell} b_{\ell+i}] \leq \sum_{i=0}^{\infty} \left(Be^{-\alpha}\right)^i = \frac{1}{1 - Be^{-\alpha}} .$$

□

LEMMA 5.2. *For every router $r$ the expected number of publish links $|r.P|$ is constant.*

*Proof.* Denote $r.level = \ell$. The probability that a node $v$ hosts an initial level-$(\ell+1)$ router $v.R_{\ell+1}$ that matches the first $\ell - 1$ bits of $u.id$ is most $B^{-(\ell-1)}$.

Further, we should consider the probability that a node emulates a shadow router of level $(\ell+1)$ with identifier matching $u.id[\ell-1]$, hence $r$ also has a publish link to it. Using the same arguments as in the proof of Lemma 5.1 above, for $0 \leq i \leq \ell$, the probability that a node $v$ has a level-$(\ell+1-i)$ router with identifier-prefix $r.id[\ell-1-i]$ and needs to emulate a level-$(\ell+1)$ shadow router with prefix $id[\ell-1]$ (i.e., emulate recursively to depth $i$) is bounded by $B^{-(\ell-i-1)}e^{-i\alpha}$.

The total probability that a node hosts a level-$(\ell+1)$ router (real or shadow) matching $u.id[\ell-1]$ is bounded by

$$\sum_{i=0}^{\infty} \frac{1}{B^{\ell-1}} \left(Be^{-\alpha}\right)^i = \frac{1}{B^{\ell-1}} \frac{1}{(1 - Be^{-\alpha})} .$$

Hence, the expected number of nodes among the $\alpha B^{\ell+d+5}$ nodes that match this criterion is bounded by

$$E[|u.P|] \leq \alpha B^{\ell+d+5} \frac{1}{B^{\ell-1}} \frac{1}{(1 - Be^{-\alpha})} = \frac{\alpha B^{d+6}}{1 - Be^{-\alpha}} .$$

□

As an immediate consequence of the above two lemmata, we get the following theorem.

THEOREM 5.1. *The expected degree of all nodes is $O(M) = O(\log n)$.*

COROLLARY 5.1. *The expected number of reference pointers for each object is $O(M) = O(\log n)$*

**5.2 Stretch** In this section we show that the worst case stretch of the lookup operation is $1+\varepsilon$. For the analysis of a lookup path, we denote the first node performing a lookup of an object *obj* by $s$, and the (closest) target node containing *obj* by $t$.

Denote the sequence of nodes steps taken by the routing algorithm as $x_1, x_2, x_3, \ldots$, where $x_1 = s$. Denote the relevant routers as $x_1.r, x_2.r, x_3.r, \ldots$ where $x_i.r.level = i$ and $x_i.r.id[i-1] = H(obj)[i-1]$. Similarly, let the sequence of publishing nodes taken from $t$ be $t = w_1, w_2, w_3, \ldots$, and the sequence of relevant routers be $t = w_1.r, w_2.r, w_3.r, \ldots$. Hence, $w_i.r.level = i$ and $w_i.r.id[i-1] = H(obj)[i-1]$. Note that some nodes may repeat within this sequence due to shadow-router emulation. For ease of notation, below we denote by $x_0 = x_1 = s$.

LEMMA 5.3. *For every $i \geq 1$, $x_i \in A_i(x_{i-1}) \subseteq A_{i+1}(s)$ and similarly, $w_i \in A_i(w_{i-1}) \subseteq A_{i+1}(t)$.*

*Proof.* By induction on $i$. For $i = 1$ we have $s = x_1$. Assume by induction that $x_{i-1} \in A_i(s)$. If $x_i.r$ is emulated then $x_i = x_{i-1}$ and we are done. Otherwise, by Lemma 2.1 (ii), $A_{i+1}(s) \supseteq A_i(x_{i-1})$. By construction, $x_i \in A_i(x_{i-1})$, and hence, $x_i \in A_{i+1}(s)$. (The case of $w_i$ and $t$ is identical). □

COROLLARY 5.2. *For every $i \geq 1$, the total distance of the path from $s = x_1$ through $x_i$ is at most*

$$\frac{\gamma}{\gamma-1} a_{i+1}(s)$$

*Proof.* By Lemma 5.3 for every $1 < j \leq i$, $x_j$ is in the ball $A_j(x_{j-1})$ that is fully contained in the ball $A_{(j+1)}(s)$, hence its radius is at most $a_{(j+1)}(s)$, and therefore $c(x_{j-1}, x_j) \leq a_{(j+1)}(s)$.

By Lemma 2.1(iii), $a_{j+1}(s) \leq \gamma^{-(i-j)} a_{i+1}(s)$. Hence, the total distance of the path from $x_1$ through $x_i$ is at most

$$\begin{aligned}\sum_{j=1}^{i-1} c(x_j, x_{j+1}) &\leq \sum_{j=1}^{i} a_{j+1}(s) \\ &\leq \sum_{j=0}^{i-1} \gamma^{-j}\ a_{i+1}(s) \\ &\leq \frac{\gamma}{\gamma-1} a_{i+1}(s)\ .\end{aligned}$$

□

LEMMA 5.4. *Let $k$ be the first index such that $s \in A_{k+d+2}(t)$ then $x_k$ contains a reference to obj.*

*Proof.* From Lemma 5.3, $x_k \in A_{k+1}(s)$. Applying Lemma 2.1(ii) on $s \in A_{k+d+2}(t)$ gives $A_{k+d+2}(s) \subseteq A_{k+d+3}(t)$. Now, from Lemma 5.3, $w_{k-1} \in A_k(t)$. Applying Lemma 2.1(i) on $w_{k-1} \in A_{k+d+3}(t)$ gives $A_{k+d+3}(t) \subseteq A_{k+d+4}(w_{k-1})$. Combining the above $x_k \in A_{k+d+2}(s) \subseteq A_{k+d+3}(t) \subseteq A_{k+d+4}(w_{k-1})$. Router $w_{k-1}.r$ has publish links such that it publishes a reference for object *obj* in all the nodes containing a level-$k$ router, whose identifier matches the prefix $w_{k-1}.r.id[k-1]$, within the ball $A_{k+d+4}(w_{k-1})$. Thus, $x_k$ must contain a reference of the type "$obj; w_{k-1}$". □

Using Lemma 5.4, we know that when the lookup path reaches $x_k$, it proceeds to $w_{k-1}, \ldots, w_1 = t$. It is left to see what is the total distance of the route $x, s = x_1, x_2, x_3, \ldots, x_k, w_{k-1}, w_{k-2}, \ldots, w_1 = t$.

THEOREM 5.2. *The stretch of the path from $s$ to $t$ is $1+\varepsilon$.*

*Proof.* The first phase of the route is the path from $s = x_1$ to $x_k$. We now make use of the assumption that $s \notin A_{k+d+1}(t)$, so $N(s, 2c(s,t)) \supseteq A_{k+d+1}(t)$ hence $A_{k+d+1}(s) \subseteq N(s, 2c(s,t))$, because of node count, and thus $a_{k+d+1}(s) \leq 2c(s,t)$. With Corollary 5.2

$$\begin{aligned}\sum_{j=1}^{k-1} c(x_j, x_{j+1}) &\leq \frac{\gamma}{\gamma-1} a_{k+1}(s) \\ &\leq \frac{2\gamma^{1-d}}{\gamma-1} c(s,t)\ .\end{aligned}$$

The second phase is the hop from $x_k$ to $w_{k-1}$. Using the triangle inequality $c(x_k, w_{k-1}) \leq a_{k+1}(s) + c(s,t) + a_k(t) \leq (2\gamma^{-d} + 1 + \gamma^{-d-1}) c(s,t)$ .

The third and last phase of the route is the traversal from $w_{k-1}, w_{k-2}$... back to $w_1 = t$. Since $a_{k+d+1}(t) \leq c(s,t)$ and from Corollary 5.2

$$\sum_{j=1}^{k-2} c(w_j, w_{j+1}) \leq \frac{\gamma^{-d}}{\gamma-1} c(s,t)\ .$$

The theorem is proven by choosing $d = O(\log(\frac{1}{\varepsilon}))$ such that

$$\varepsilon \geq \frac{1}{\gamma^d}\left(\frac{2\gamma}{\gamma-1} + 2 + \frac{1}{\gamma} + \frac{1}{\gamma-1}\right)\ . \tag{5.2}$$

□

## 6 A Two-tier Construction

Almost all the file sharing systems used currently in the Internet, e.g., Kazaa, Gnutella2, Fast Track, and

others, use a hierarchical structure. Nodes are divided into nodes and super-nodes. While all nodes participate in forwarding and returning lookup requests, only the super nodes (whose reliability is higher) maintain the data itself and answer the lookup queries.

In this section, we briefly sketch an alternative approach that enables to embrace the asymmetrical structure of real networks, and captures it in a two-tier architecture. The classification into the two tiers allows to take into account the difference between nodes in a number of domains: communication speed, persistence, CPU power and storage capacity, and so on. The main difference is that only supernodes store objects, whereas regular nodes only assist in routing queries to supernodes.

This distinction leads to a dramatic improvement in the lookup algorithm. Our two-tier construction employs only a constant expected number of out-going links from all regular nodes. As noted in several DHT works, e.g., in [18, 14, 1, 15, 8, 5, 16], the number of links maintained by each node affects the adaptability of the network to changes. Thus, this is an important improvement over previous locality-aware DHTs which employ a logarithmic number of out-links per node [17, 4, 22, 3].

More formally, we assume that there are two types of nodes in the system, regular nodes and supernodes. Denote the set of regular nodes $R$, supernodes $S$ , all nodes $V = R \bigcup S$, the number of all nodes $|V| = n$. Each regular node hosts only one initial router, chosen from among $R_1, ..., R_{(M+1)}$ arbitrarily. Supernodes always host $R_1$ in addition.

In order to make up for the fact that finding a node hosting a router with certain characteristics becomes $M$ times harder, we redefine $A_i(v)$ to be the smallest neighborhood around $v$ containing $\alpha B^i M$ nodes. The definitions of neighbor and publish links carry identically, with the distinction that they make use of the new neighborhood definitions.

Initiating a search path requires an additional stage. In this stage, a node hosting a level-1 router ($R_1$) is sought. Once such a node is reached, the digit-fixing phase of routing can commence. In order to find a level-1 router, each node maintains one additional type of link, *closest*, which connect to the closest super-node in its vicinity.

Analysis similar to the above shows the following properties. In the two-tier construction, finding a target $t$ from an initiating node $x$ incurs a stretch of at most $1 + \varepsilon$ from the $x.closest$ to $t$ (this part is proven just as before); and an additional stretch-1 hop from $x$ to $x.closest$. The total stretch is therefore bounded by $2+\varepsilon$. The construction uses a constant expected number of outgoing links per node. The incoming degree of nodes are affected by the overall density of supernodes among regular ones. The density of super-nodes can be controlled by system policy in various ways. Assuming that $|R| = m|S|$ and that dispersal of super-nodes is uniform in the space, then the in-degree of regular nodes is also expected constant, and that of super-nodes is logarithmic.

Full details of the two-tier construction appear in [2], and will be included in the full version of the paper.

## 7 Dynamic Node Arrivals and Departures

In this section we sketch how nodes may dynamically arrive and depart from the system.

We assume that once two nodes $x, y$ connect (by $x$ sending a message that arrives to $y$) they may exchange messages and discover the real distance $c(x, y)$ between them.

**7.1 Node arrival** When a new node arrives to the system it needs to do several things: (1) acquire an id for each of its routers, (2) establish network links for each of its routers, (3) acquire necessary object references.

**Acquiring identifier for each router** Each node chooses for each initial router, $R_1 \dots R_{(M+1)}$, an identifier of $M$ uniformly independent random radix $B$ digits. Note that due to a significant change in the number of nodes, the parameter $M$ may change. In such a case, routers may need to add a new digit to each of their identifiers.

**Finding the nearest neighbor** As part of the process of establishing router links, a node first needs to identify the closest neighbor it has in the network. Hildrum, Kubiatowicz and Rao [7] propose to use the PRR routing scheme in a backward manner, in order to locate the nearest neighbor with high probability in PRR like networks. As the authors note in their conclusion, it is possible to combine the techniques of [7] with the LAND construction. Using the basic LAND architecture a load balanced distributed nearest neighbor search will take an expected logarithmic number of steps. The nearest neighbor is always found, unlike [7] which has only high probability guaranties.

Algorithm FIND-CLOSEST for a node $x$ is as follows. Let $u$ be any node in the network known to $x$, e.g., an initial contact point. We denote $w_{M+1} = y$, and its level-$(M + 1)$ initial router by $w_{M+1}.r = w_{M+1}.R_{M+1}$. For $\ell = M + 1$ down to 2, take from among all incoming links into $w_\ell.r$ the router $w_{\ell-1}.r$ closest to $x$. By construction, $w_{\ell-1}.r$ is a level-$(\ell - 1)$ router with $w_{\ell-1}.r.id[\ell-2] = w_{(M+1)}.r.id[\ell-2]$. At the end, set the closest node known to $x$, denoted $x.closest$ to $w_1$.

The FIND-CLOSEST algorithm is depicted in pseudo-

code in Figure 3.

A node $v$ that wants to find its closest neighbor invokes $v.closest = v.R_{(M+1)}$.FIND-CLOSEST$(v, M+1)$.

FIND-CLOSEST$(v, \ell)$ at router $u.r$:
  If $\ell = 1$ send $u$ to $v$ and return;

  Let $S_i$ denote the set of all incoming links into $u.r$;
  Let $w$ among $S_i$ be the closest to $v$;
  $w.r$.FIND-CLOSEST$(v, \ell - 1)$;

Figure 3: The FIND-CLOSEST algorithm.

**Establishing network links** Once the id and level of a router is set, and the closest node to the node hosting it is known, the router is left with the task of establishing links as defined in Section 4.

For a router $v.r$ with level $\ell$, the main difficulty is to find all the level $\ell+1$ routers with prefix $v.r.id[\ell-1]$ in the ball $A_{\ell+d+5}(v)$. Router $v.r$ also needs to inform all routers $u.r$ of level $\ell-1$ with prefix $v.r.id[\ell-2]$ such that $v \in A_{\ell+d+4}(u)$. This can be done, again, by finding all routers $u.r$ in $A_{\ell+d+5}(v)$ with prefix $v.r.id[\ell-2]$.

The LOCATE algorithm for a router $v.r$ of level $\ell$ is as follows. Let $s$ be the closest node to $v$. For every combination of digits $b_1, b_2 \in [0..B-1]$, route from $s$ to a level-$(\ell+2)$ router $u.r$ such that $u.r.id[\ell+1] = v.r.id[\ell-1]||b_1||b_2$. This routing is done in an identical manner to the routing phase of LOOKUP in Figure 2, i.e., using the $L(i)$ links. Let $Y$ denote the set of routers $u.r$ reached by this procedure. Let $S(\ell+3)$ be the set of level-$(\ell+3)$ routers that appear as publish links of routers in $Y$, i.e., $S(\ell+3) = \bigcup_{y \in Y} y.P$. Obtain $S(\ell+2)$ by taking all incoming publish links into $S(\ell+3)$ from routers of level $\ell+2$. Then recursively, obtain $S(\ell+1)$ by taking all publish links going into $S(\ell+2)$. And so on, until we have $S(\ell-1)$. From $S(\ell+1)$, router $v.r$ selects neighbor links whose distance from $v$ does not exceed $a_\ell$, and keeps publish links whose distance from $v$ does not exceed $a_{\ell+d+5}$. Then $v$ informs nodes in $S(\ell-1)$ about its arrival.

The LOCATE algorithm in depicted in pseudo-code in Figure 4.

### 7.2 Node departure

When a regular node $x$ of level $\ell$ leaves the network, the level $\ell-1$ nodes whose neighbor link contained a router $x.r$ need to be updated, and $x.r$ removed from their list. If $x.r$ was a neighbor link of a router $v.r$, then $v.r$'s next closest publish link becomes the neighbor link, unless this link is too far away in which case $v$ emulates a shadow node. The links for this emulation are acquired using the LOCATE algorithm.

LOCATE at router $v.r$ of level $\ell$:
  for every combination of digits $b_1, b_2 \in B$
    $v.closest.R_1$.SEARCH$(v.r.id[\ell-1]||b_1||b_2, \ell, v)$;
  wait for replies, accumulate in $S$;
  set $v.r.L(i) = argmin_{u.r \in S}\{c(u,v) \mid u.r.level = \ell+1 \ \wedge$
    $u.r.id[\ell] = v.id[\ell-1]||i \ \wedge \ c(v,u) \le a_\ell\}$ ;
    // emulate $v.r.L(i)$ if empty
  set $v.r.P = \{u.r \in S \mid u.r.level = \ell+1 \wedge$
    $u.r.id[\ell-1] = v.r.id[\ell-1] \wedge c(v,u) \le a_{\ell+d+5}\}$;
  inform all level-$(\ell-1)$ routers in $S$ about $v$'s arrival;

SEARCH$(prefix, \ell, v)$ at router $u.r$:
  if $u.r.level = \ell+1$
    INLINKS$(v, 3)$;
  else $u.r.L(prefix_{u.level})$.SEARCH$(prefix, \ell, v)$;

INLINKS$(v, j)$ at router $u.r$
// recurse for $j$ levels searching for in-links
  let $I$ denote the incoming publish links of $u.r$;
  send $v$ the set $I$;
  if $(j > 1)$
    for each $w.r \in I$ : $w.r$.INLINKS$(v, j-1)$;

Figure 4: The LOCATE algorithm.

### 7.3 Analysis of dynamic algorithms

LEMMA 7.1. *For a node $x$, Algorithm* FIND-CLOSEST *finds the closest neighbor of $x$.*

*Proof.* Denote by $u$ the node at which FIND-CLOSEST is initiated by $x$. Denote $w_{M+1}.r = u.R_{M+1}$, and denote the sequence of routers traversed by the algorithm by $w_{(M+1)}.r, w_M.r, ..., w_1.r$. Let $s$ denote the closest node to $x$ in the network.

We show by induction backward from $M+1$ to 1 that $w_i \in A_{i+2}(x)$ and $w_i.r$ is the closest router to $x$ which satisfies $w_i.r.id[i-1] = w_{(M+1)}.r.id[i-1]$. The base obviously holds. We now prove the induction step.

Consider the routing path from $s$ to $w_{(M+1)}r..id$ using the regular digit-fixing routing method. Denote the router reached in the $i$'th routing step by $y_i.r$, i.e., $y_i.r.level = i$ and $y_i.r.id[i-1] = w_{(M+1)}.r.id[i-1]$. By Lemma 5.3, $y_i \in A_{i+1}(s)$. Since $s \in A_1(x)$, by Lemma 2.1(ii), $A_{i+1}(s) \subseteq A_{i+2}(x)$, and so $y_i \in A_{i+2}(x)$. Hence, if there is any level-$i$ router $c_i.r$ closer to $x$ than $y_i.r$, then $c_i \in A_{i+2}(x)$ as well. By the induction hypothesis, $w_{i+1} \in A_{i+3}(x)$, hence, by Lemma 2.1(i), $A_{i+2}(x) \subseteq A_{i+3}(x) \subseteq A_{i+4}(w_{i+1})$. From Lemma 2.1(i), this means $A_{i+4}(w_{i+1}) \subseteq A_{i+5}(c_i)$, and therefore, $c_i.r$ has a publish pointer to $w_{i+1}.r$. Therefore, Algorithm FIND-CLOSEST would find $c_i.r$ in this step, and in fact, $w_i.r = c_i.r$. This completes the proof. □

LEMMA 7.2. *For a router $x.r$ with level $\ell$, Algorithm* LOCATE *finds all the appropriate level-$(\ell+1)$ routers in $A_{\ell+d+5}(x)$ and all appropriate routers $w.r$ with level $\ell-1$ such that $x \in A_{\ell+d+4}(w)$.*

*Proof.* Let $s$ be the closest node. By applying Lemma 2.1, $A_{\ell+d+5}(x) \subseteq A_{\ell+d+6}(s)$. By Lemma 5.3 each router $y.r$ that is routed to in algorithm LOCATE is in $A_{\ell+3}(s) \subseteq A_{\ell+d+6}(s)$. In addition, each such router has level $\ell+2$ so its publish links cover all appropriate level $\ell+3$ routers in $A_{\ell+2+d+5}(y)$. Applying Lemma 2.1 again we get $A_{\ell+d+5}(x) \subseteq A_{\ell+d+6}(s) \subseteq A_{\ell+d+7}(y)$. Hence, by following incoming links backwards down to level $\ell-1$, all appropriate level-$(\ell-1)$ routers within $A_{\ell+d+5}(x)$ are guaranteed to be found.

For every $w$ such that $x \in A_{\ell+d+4}(w)$, we have that $A_{\ell+d+4}(w) \subseteq A_{\ell+d+5}(x)$. By a similar reasoning to the above, we find all such $w$'s. □

LEMMA 7.3. *For a node arrival:*
*(i) The expected number of nodes that change their state is logarithmic.*
*(ii) The expected number of messages sent is $O(M^2)$*

*Proof.* For each $\ell \in [1 \ldots M]$, routing to each of the level $\ell+2$ routers takes $\ell+2$ messages.

Once such a router is reached, a message is sent to each of its links. The expected number of links is constant (see Theorem 5.1), and all choices are independent. Thus recursively finding all appropriate links for all $M$ routers will cause sending an expected $O(M)$ number of messages.

The number of nodes that change their state for each $\ell \in [1 \ldots M]$ is the number of appropriate routers in $A_{\ell+d+5}(x)$, the expected number of these routers is constant. □

## 8 Conclusions

LAND is the first peer-to-peer network and lookup algorithm that has worst case $1+\varepsilon$ stretch. The network is a variation on the access scheme of [17], yet whereas PRR achieves an expected (high) constant stretch, our construction yields any desired stretch in the worst case.

The two-tier variant of LAND embraces current file-sharing architectures, in which data is stored only on super-nodes. Two-tier LAND is the first constant-degree locality-aware DHT.

In the future, we plan to explore in full the following extensions of our scheme.

First, it is possible to make use of our approach with different parameters, e.g., trade higher degree for lower diameter, while keeping the goal of guaranteed low stretch.

Second, our router-hosting abstraction leads to an intuitive yet powerful form of fault tolerance. Specifically, the number of initial routers hosted by a node may be increased, and accordingly, links may be replicated, in order to achieve fault tolerance with locality guarantees.

Finally, the deployment of algorithms such as [17] and LAND remains an active area of research, e.g., in [22, 4]. We envision that many of the insights and optimizations in these systems may be applied for a real-life implementation of LAND.

## References


[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal D. Malkhi and E. Pavlov. "A Generic Scheme for Building Overlay Networks in Adversarial Scenarios". In *International Parallel and Distributed Processing Symposium* (IPDPS 2003), April 2003, Nice, France.

[2] I. Abraham, D. Malkhi and O. Dobzinski . "LAND: Locality Aware Networks for Distributed Hash Tables". Technical Report 2003-75, Leibnitz Center of the School of Computer Science and Engineering, the Hebrew University of Jerusalm, July 2003.

[3] R. Chand and P. Felber. "A Scalable Protocol for Content-Based Routing in Overlay Networks". In IEEE International Symposium on Network Computing and Applications (NCA'03), Cambridge, MA, April 2003.

[4] P. Drushel and A. Rowstron. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". *Proceeding of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware),* November 2001.

[5] P. Fraigniaud and P. Gauron. "The Content-Addressable Network D2B". Technical Report 1349, LRI, Univ. Paris-Sud, France, January 2003.

[6] C. Gavoille and M. Gengler. Space-efficiency for routing schemes of stretch factor three. *Journal of Parallel and Distributed Computing*, 61(5):679–687, 2001.

[7] K. Hildrum, J. Kubiatowicz and S. Rao Another Way to Find the Nearest Neighbor in Growth-Restricted Metrics UC Berkeley, Computer Science Division Tecnical Report UCB/CSD-03-1267, August, 2003.

[8] F. Kaashoek and D. R. Karger. "Koorde: A Simple Degree-optimal Hash Table". In *2nd International Workshop on Peer to Peer Systems* (IPTPS '03), February 2003, Berkeley, CA.

[9] D. Karger, E. Lehman, F. T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web". *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 654–663, May 1997.

[10] D. R. Karger and M. Ruhl. "Finding Nearest Neighbors in Growth-restricted Metrics". ACM Symposium on Theory of Computing (STOC '02), Montreal, May 2002.

[11] X. Li and C. G. Plaxton. "On name resolution in peer-to-peer networks." In *Proceedings of the 2nd ACM Worskhop on Principles of Mobile Commerce* (POMC), pp. 82–89, October 2002.

[12] N. Lynch, D. Malkhi and D. Ratajczak. "Atomic data access in distributed hash tables". In *Proceedings of the International Peer-to-Peer Symposium*, March 2002.

[13] D. Malkhi. "Dynamic Lookup Networks: A position paper". In *Proceedings of the International Workshop on Future Directions in Distributed Computing* (FuDiCo), LNCS Volume 2584, pp. 93–96. Bertinoro, Italy, 2002,

[14] D. Malkhi, M. Naor and D. Ratajczak. "Viceroy: A scalable and dynamic emulation of the Butterfly". In *Proceeding of the 21 st ACM Symposium on Principles of Distributed Computing (PODC'02)*, July 2002.

[15] G.S Manku, M. Bawa, P. Raghavan. "Symphony: Distributed Hashing In Small World". In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003.

[16] M. Naor and U. Wieder. "Novel Architectures for P2P Applications: the Continuous-Discrete Approach". In *The Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '03)*, 2003.

[17] C. Plaxton, R. Rajaraman, and A. Richa. "Accessing nearby copies of replicated objects in a distributed environment". *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures* (SPAA 97), pp. 311–320, June 1997.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. "A scalable content-addressable network". In *Proceeding of the ACM SIGCOMM 2001 Technical Conference.* August 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for Internet applications". In *Proceedings of the SIGCOMM 2001*, August 2001.

[20] M. Thorup and U. Zwick. Approximate distance oracles. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC '01)*, pages 183–192. ACM Press, 2001.

[21] M. Thorup and U. Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures (SPAA '01)*, pages 1–10. ACM Press, 2001.

[22] B.Y. Zhao, J. D. Kubiatowicz and A.D. Joseph. "Tapestry: An infrastructure for fault-tolerant wide-area location and routing". U.C. Berkeley Technical Report UCB/CDS-01-1141, April, 2001.