Principles of Locality-Aware Networks for Locating Nearest Copies of Data

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

November 25, 2003

Abstract

Building overlay network tools for locating information in a manner that exhibits locality-awareness is crucial for the viability of large internets. It means that costs are proportional to the actual distance of interacting parties, and in many cases, that load may be contained locally. This paper presents a step-by-step decomposition of several locality-aware networks, that support distributed content-based location services. It explains their common principles and their variations with simple and clear analysis.

1 Introduction

Problem statement. This paper considers the problem of forming an overlay routing network that locates objects placed in arbitrary network locations. More formally, the network constitutes a metric space, with a cost function $c(x, y)$ denoting the "distance" from x to y. Let $s = x_0, x_1, ..., x_k = t$ be the path taken by the search from a source node s to the object residing on a target node t. The main design goal to achieve is constant *stretch*. Namely, that the ratio $\frac{c(x_0,x_1)+...+c(x_{k-1},x_k)}{c(s,t)}$ is bounded by a (small) constant. Another important goal is to keep node degree low, so as to prevent costly reconfigurations when nodes join and depart. Thus, trivial solutions that connect all nodes to each other are inherently precluded.

Two variants of the problem may be considered. In one, the object of interest may be placed on any desired target node t by the algorithm designer. A primary application of this problem model is a distributed reference directory such as a Distributed Hash Table (DHT). In the second variant, the object's location is uncontrolled, e.g., a distributed web cache, or a location service for mobile device, and the goal is to locate the closest copy of the object. The difference between these two variants in location algorithms is that in the former, the target's location can be a virtual address determined by the algorithm's designer; in the latter, a virtual address can only be used for publishing the true object location. The lookup protocol routes towards the virtual target, until it coincides with a published reference of the real object location. In terms of the locality issues of importance to us, these problems are very similar, and we consider the slightly harder problem of finding nearest copies of data.

Previous work. The problem of forming overlay routing networks was considered by several recent works in the context of networks that are searchable for content. Many of the prevalent overlay networks were formed for routing search queries in peer-to-peer applications, and exhibit **no** locality awareness.

There are several known schemes that provide locality awareness, including [10, 11, 12, 9, 2]. All of these solutions borrow heavily from the PRR scheme [10], yet they vary significantly in their assumptions and properties. Some of these solutions are designed for a uniform density space [9]. Others work for a class of metrics space whose growth rate is bounded both from above and from below [10, 11, 12, 6], while others yet cope with an upper bound only on the growth rate [2]. There is also variability in the guarantee provided on the stretch: In [9], there is no bound on stretch (except the network diameter). In [10], the stretch is an expected constant, a rather large one which depends on the growth bound. And in [2], the stretch can be set arbitrarily small $(1+\varepsilon)$. Diversity is manifested also in the node degree of the schemes.

It is worth noting what would happen if we remove the growth-bound restriction altogether. In the theory of networks design, a related problem of *name independent routing* has been considered as early as 1989 [4]. In this problem, we are given a network $G = \langle V, E, \omega \rangle$, where V is a set of nodes, E are edges, and ω is an edge-cost function. Node names are chosen as an arbitrary permutation on $\{1, ..., n\}$. A routing request destination is given in the form of a node name, and the goal is to route with constant stretch. A trivial solution keeps full optimal routing tables at each node with memory cost of $O(n \log n)$. Hence, the challenge is in devising *compact* schemes whose memory is $o(n \log n)$. A compact solution for name independent routing can directly support a locality aware object lookup service or a DHT, with node degree induced from the routing-table requirements. However, for general metric spaces, routing with stretch < 3 requires a routing table of size $O(n)$ per node [5], and the best known schemes [3, 1] achieve stretch 5 and 3, respectively, with $O(\sqrt{n})$ entries per node.

A step-by-step deconstruction. This paper offers a deconstruction of the principles that underlie these locality-aware schemes step by step, and indicate how and where they differ. It demonstrates the principles of locality awareness in a simplistic, yet reasonable (see [6]) network model, namely, a network with *power law* latencies. Finally, it provides guidelines for extending to growth-bounded metrics.

The logical steps we make are the following. First, we build *geometric routing*, whose characteristic is that the routing steps toward a target increase geometrically in distance. This is achieved by having large flexibility in the choice of links at the beginning of a route, and narrowing it down as the route progresses. Geometric routing alone yields a cost which is proportional to the network diameter. The designs in [9, 6] make use of it to bound their routing costs by the network diameter.

The next step is unique to the design of LAND in [2]. Its goal is to turn the expectation of geometric routing into a worst-case guarantee. This is done while increasing node degree only by a constant expected factor. The technique to achieve this is for nodes to *emulate* links that are missing in their close vicinity as *shadow nodes*. In this way, the choice of links **enforces** a distance upper bound on each stage of the route, rather than probabilistically maintaining it. If no suitable endpoint is found for a particular link, it is emulated by a shadow node.

The final step in our deconstruction describes how to bring down routing costs from being proportional to the network diameter (which could be rather large) to being related directly to the actual distance of the target. This is done via a technique suggested by Plaxton et al. in [10], that

makes use of short-cut links that increase the node degree by a constant factor. With a careful choice of the short-cut links, as suggested by Abraham et al. in [2], this guarantees an optimal stretch, irrespective of the growth-rate parameters of the network.

We supplement our exposition with two enhancements. In Section 4, we sketch how to adapt the building blocks we present to a growth-bounded metric space. In Section 5 we provide intuition on dynamic maintenance components.

The analysis we present throughout is simple and proofs are short and intuitive. In our belief, the simplicity and the elegance of analysis may lead to improved practical deployments of locality-aware schemes.

2 Preliminaries

The system consists of a set V of N nodes. 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 $\langle V, c\rangle$ forms a metric space. From here on, we refer to the cost as the *distance* between nodes.

Density. The set of nodes within distance r from x is denoted $N(x, r)$. A *power law latency* assumption is expressed as follows:

$$|N(x, r)| = \Gamma r^p \ ,$$

for some known constants $\Gamma \geq 2, p \geq 1$.[1] In the calculation below, we take $p = 2$.

For convenience, we define neighborhoods $A_k(x) = N(x, 2^k)$, and the radius $a_k = 2^k$. Thus, we have that $|A_k(x)| = \Gamma 4^k$.

Routing entities and identifiers. For the purpose of forming a routing structure among nodes, nodes need to have addresses and links. We refer to a *routing entity* of a node as a router, and say that the node *hosts* the router. Thus, each node u hosts an assembly of routers.

Each router $u.r$ has an identifier denoted $u.r.id$, and a level $u.r.level$. Identifiers are chosen uniformly at random. The radix for identifiers is selected for convenience to be 4. This is done so that a neighborhood of radius 2^k shall contain in expectation constant number of routers with a particular length-k identifier. Indeed, the probability of a finding a router with a specific level and a particular prefix of length k is $1/4^k$. According to our density assumption, a neighborhood of diameter 2^k has $\Gamma 4^k$ nodes. Hence, such a neighborhood contains in expectation Γ routers matching a length-k prefix.

Let $M = \log_4 N$. Identifier strings are composed of M digits. The level is a number between 1 and M. A level-k router has links allowing it to 'fix' its k'th identifier digit. Routers are interconnected in a butterfly-like network, such that level-k routers are linked only to level-$(k+1)$ and level-$(k-1)$ routers.

Let d be a k-digit identifier. Denote $d[j]$ as the prefix of the j most-significant digits, and denote d_j as the j'th digit of d. A concatenation of two strings d, d' is denoted by $d||d'$.

[1] In fact, our simple analysis can be easily adapted to accommodate an assumption $\Gamma r^p \leq |N(x, r)| \leq \Delta r^p$, or similarly, an assumption that nodes are placed uniformly at random in space. For brevity, we use the above assumption.

Objects and Hashing. Let $\mathcal{A}$ be the set of searchable objects in the system. We make use of a uniformly distributed hash function H on object names. For any $A \in \mathcal{A}$, reference information about A's location is stored on a node whose identifier has the longest matching prefix to $H(A)$.

Summary of notations. For convenience, here is a short summary of constants and notations.
- N denotes the number of nodes in the network, $M = \log_4 N$.
- Router r hosted by node v is denoted $v.r$, its identifier is $v.r.id$, and its level $v.r.level$.
- The set of nodes within distance r from x is denoted $N(x, r)$.
- $A_i(x) = N(x, 2^i)$, and the radius is $a_i = 2^i$.

3 The routing network

3.1 Step 1: Geometric routing

Every node v initially hosts $M + 1$ routers, denoted $R_1, ..., R_{M+1}$. For $k = 1..M$, a level-k router r hosted on a node v maintains outgoing *neighbor* links as follows.

neighbor: There are four neighbor links, denoted $L(b)$, $b \in \{0..3\}$. Each one of the links $L(b)$ is selected as the closest node within $C_b(r)$, where $C_b(r) = \{u \in V \mid \exists s, u.s.id[k] = v.id[k-1]||b, u.s.level = k+1\}$. The link $L(b)$ 'fixes' the k'th bit to b, namely, it connects to the closest node that has a level-$(k+1)$ router whose identifier matches $v.R_k[k-1]||b$.

Publish and lookup. As there is no relationship between the location of an object *obj* and its identifier, we use a 'home' node to store a reference to the location of *obj*. In fact, in order to maintain locality, we distribute reference information on *obj* in a number of home nodes dispersed in the network. These are nodes u, such that $u.R_k.id$ matches a $(k-1)$-length prefix of $H(obj)$.

More precisely, a node t that holds a copy of an object *obj* publishes this fact as follows. Starting from $t.R_1$, we follow neighbor links so as to fix target bits according to $H(obj)$. Let $w_1 = t$, and denote by $w_1.r = t.R_1$. From a level-k router $w_k.r$, $k \leq M$, reached in the k'th step, we follow $w_k.r.L[H(obj)_k]$ to reach $w_{k+1}.r$. Thus, $w_{k+1}.r$ is a level-$(k+1)$ router satisfying $w_{k+1}.r.id[k] = H(obj)[k]$. Each node w_{k+1} along the path retains a reference to *obj* of the form $obj; w_k$.

The lookup of an object *obj* residing on a node t from an origin node s proceeds as follows. Starting with $x_1 = s$, using router $x_1.r = s.R_1$, we move from a router $x_k.r$ to $x_{k+1}.r$ using the level-k router links, fixing the k'th bit to that of $H(obj)$ in the k'th step. The loop goes as follows: So long as the target was not found, then from the current router $x_k.r$, first check if there is a reference of the form $obj; w_{k-1}$ on x_k. If so, move to it. Otherwise, if $H(obj)_k$ is b, continue at $x_k.r.L(b)$.

Analysis of expected stretch. As already mentioned above, for any given length-k binary prefix $I[k]$, the ball $A_k(v)$ contains in expectation at least one node u with $u.R_{(k+1)}.id[k] = I[k]$. Hence, the expected distance of the link $v.R_k.L(b)$, for $b \in [0..3]$, is at most $a_k = 2^k$. Further, the expected cumulative length of a k-hop path is dominated by the last step, and is bounded by $\sum_{i=1..k} 2^i \leq 2^{k+1}$. The good news then are that any search path is bounded in expectation by a constant stretch over the network diameter, namely, by 2^{n+1}.

The bad news are that as this is an analysis of the expected case only, in the worst case, routes may be much longer. Our next enhancement puts a worst-case bound on the distance of links, and consequently, on the total length of any path.

3.2 Step 2: Shadow routers

The idea of bounding the distance of links is very simple: If a link does not exist within a certain desired distance, it is *emulated* as a shadow router. More precisely, for any level $1 \leq k \leq M$ let r be a level-k router hosted by node v (this could itself be a shadow router, as described below). For $b \in [0..3]$, if $C_b(v)$ contains no node within distance 2^k, then node v emulates a level-$(k+1)$ *shadow router* s that acts as the $v.r.L(b)$ endpoint. Router s's id is $s.id = v.r.id[k-1]||b$ and its level is $(k+1)$.

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 $C_j(s)$ within distance 2^{k+1}. In such a case v also emulates a shadow router that acts as the $s.L(j)$ endpoint.

Emulation continues recursively until all links of all the shadow routers emulated by v are found (or until the limit of M levels is reached).

Analysis of node degree with shadow routers. With shadow routers, we have a deterministic bound of 2^k on the k'th hop of a path, and a bound of $\sum_{i=1..k} 2^i = 2^{k+1}$ on the total distance of a k-hop path.

A different concern we have now is that a node might need to emulate many shadow routers, thus increasing the node degree. The following lemma tells us that in expectation, each router incurs only a constant number of emulated routers on its behalf.

Lemma 3.1 *For every router r hosted by a node v, the expected number of shadow routers hosted by v to emulate r's links is constant.*

Proof: Recall that for every ℓ, $|N(v, 2^\ell)| = \Gamma 4^\ell \geq 2 \cdot 4^\ell$. Let s be any router hosted by v whose level is ℓ. The probability that the i'th neighbor link of s is found inside $A_\ell(v)$ is at least $1 - \left(1 - \frac{1}{4^\ell}\right)^{2 \cdot 4^\ell} \geq 1 - e^{-2}$.

Let $k = r.level$ denote the level of router r. For index $0 \leq i \leq n-k$, let b_{k+i} be a random variable that counts the number of shadow routers that v recursively emulates in order to maintain links for router r.

So $b_k = 1$ and for $1 \leq i \leq n-k$, each of the b_{k+i-1} nodes has 4 neighbor links with a probability of emulating each one bounded by e^{-2}. Therefore $E[b_{k+i} \mid b_{k+i-1}] \leq b_{k+i-1}\frac{4}{e^2}$ and due to the independence of the identifiers $E[b_{k+i}] \leq E[b_{k+i-1}]\frac{4}{e^2}$. Thus by induction $E[b_{k+i}] \leq \left(\frac{4}{e^2}\right)^i$. The expected total number of shadow router emulated by node v on r's behalf is bounded by:

$$E[\sum_{0 \leq i \leq n-k} b_{k+i}] \leq \sum_{i=0}^{\infty} \left(\frac{4}{e^2}\right)^i = \frac{1}{1 - \frac{4}{e^2}} \ .$$

□

Corollary 3.2 *Let v be any node. The total expected number of routers hosted by v is $O(M)$.*

Proof: Node v hosts $M+1$ initial routers. In expectation, it hosts a constant number of shadow routers in order to maintain the links of each one of the initial routers. □

As for stretch, it is possible that the search and the publish paths from a source and a target converge towards each other, such that the k'th step of searching and the k'th step of publishing coincide. In that case, the route will have distance $O(2^k)$.

However, there is no guarantee that this happens. In the worst case, starting from a source very close to the target, the search route and the publishing route could drift away from each other until they meet at a late step. This could lead to a search route that goes a distance that is the full diameter of the network, even in the case that the source and target are initially very close to each other. Our next and final improvement fixes this problem.

3.3 Step 3: Publish links

The technique that guarantees a constant stretch is to 'publish' references to an object in a slightly bigger neighborhood than the regular links distance. The intuition on how to determine the size of the enlarged publishing-neighborhood is as follows. The route that locates *obj* on t from s is composed of three parts: (i) the path from x_1 to x_k, whose length is bounded by a_{k+1}; (ii) the path from w_{k-1} to w_1, bounded by a_k; and (iii) the hop from x_k to w_k, which is bounded by the triangle inequality by the sum of distances of x_1 from x_k, plus the distance from w_{k-1} to w_1, plus the distance from s to t. In order to achieve a stretch bound close to 1, we should therefore guarantee that a hop from x_k to w_{k-1} occurs as soon as a_k is proportional to $\varepsilon c(s,t)$. This will yield a total route distance proportional to $(1+\varepsilon)c(s,t)$.

More precisely, let $d = O(\log(\frac{1}{\varepsilon}))$ be a parameter determined as in Equation 1 below. Each router r of level k hosted by a node v has, in addition to neighbor links, the following set of links:

publish: The set of publish links, $r.P$, contains all the level-$(k+1)$ routers with the same first $k-1$ prefix identifier bits as $r.id$ within the ball $A_{k+d+4}(v)$. Formally, $r.P = C(r) \cap A_{k+d+4}(v)$, where $C(r) = \{u \in V \mid \exists s, u.s.level = k+1, u.s.id[k-1] = r.id[k-1]\}$.

Analysis of stretch. In order to bound the stretch of all search routes, denote as above by *obj* an object residing on node t, and let s be an origin searching for *obj*. Denote the publish path of *obj* from t by $t = w_1, ..., w_{n+1}$, and the corresponding routers by $w_1.r, ..., w_{n+1}.r$. Thus, $w_i.r$ is a level-i router satisfying $w_i.r.id[i-1] = H(obj)[i-1]$. Note that some nodes may repeat due to shadow router emulation. The routing steps in search of *obj* from s are $x_1.r, ..., x_k.r, w_{k-1}.r, ..., w_1.r$, where $s = x_1$ and $t = w_1$, $x_i.r.level = i$, and $x_i.r.id[i-1] = H(obj)[i-1]$.

Lemma 3.3 *For every $i \geq 1$, $w_i \in A_{i+1}(t)$, and likewise, $x_i \in A_{i+1}(s)$.*

Proof: By induction on i. For $i = 1$ we have $t = w_1$. Assume by induction that $w_{i-1} \in A_i(t)$. If $w_i.r$ is emulated by the same node as $w_{i-1}.r$ then we are done. Otherwise, by construction $w_i \in A_i(w_{i-1})$. Since $w_{i-1} \in A_i(t)$, taking the sum of radii of $A_i(t)$ and $A_i(w_{i-1})$ we get $A_i(w_{i-1}) \subseteq A_{i+1}(t)$. Putting the above together, we obtain $w_i \in A_{i+1}(t)$. The case for x_i is identical. □

Lemma 3.4 *Let k be the first index such that $t \in A_{k+d+2}(s)$. Then x_k contains a reference to obj linking to w_{k-1}.*

Proof: From Lemma 3.3, we have $x_k \in A_{k+1}(x)$ and $w_{k-1} \in A_k(t)$. Hence,

$$c(x_k, w_{k-1}) \leq c(x_k, s) + c(s, t) + c(t, w_{k-1}) \leq a_{k+1} + a_{k+d+2} + a_k \leq 2a_{k+d+2} \leq a_{k+d+3} \ .$$

Hence, $x_k \in A_{k+d+3}(w_{k-1})$. Since $w_{k-1}.r$ has level-$(k-1)$ publish links to all level-k routers with a prefix $w_{k-1}.r.[k-2] = H(obj)[k-2]$ within $A_{k+d+3}(w_{k-1})$, there is indeed a reference to obj on x_k pointing to w_{k-1}. □

Let k be the first index such that $t \in A_{k+d+2}(s)$. By Lemma 3.4, we know that when the lookup path reaches x_k, it proceeds to w_{k-1} and from there backwards to t. It is left to see what is the total distance of the route $s = x_1, \ldots, x_k, w_{k-1}, ..., w_1 = t$.

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

Proof: Since k is the first index for which $t \in A_{k+d+2}(s)$, we have that $c(s, t) \geq a_{k+d+1}$.

The total distance of the route from x_1 through x_k is bounded by $a_{k+1} \leq 2^{-d} a_{k+d+1} \leq 2^{-d} c(s, t)$. The step from x_k to w_{k-1} has distance at most $a_{k+1} + c(s, t) + a_k \leq c(s, t)(2^{-d} + 1 + 2^{-d-1})$. Finally, the route from w_{k-1} to w_1 has distance at most $a_k \leq 2^{-d-1} a_{k+d+1} \leq 2^{-d-1} c(s, t)$. It follows that the stretch is bounded by

$$2^{-d} + 2^{-d} + 1 + 2^{-d-1} + 2^{-d-1} = 1 + \frac{3}{2^d} \leq 1 + \varepsilon \ ,$$

where the last inequality is obtained by choosing

$$d \geq \log\left(\frac{3}{\varepsilon}\right) \ . \tag{1}$$

□

Analysis of the number of publish-links. We also need to show that the number of publish links per node is logarithmic, in order for the total number of links per node to remain logarithmic.

Lemma 3.5 *For every node router v, the expected number of publish links is $O(M)$.*

Proof: We prove that the expected number of links for every router r of level ℓ, for each $\ell = 1..\log N$, is a constant.

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 $\ell - 1$ digits of $u.id$ is most $4^{-(\ell-1)}$.

Further, we should consider the probability that a node emulates a shadow router of level $(\ell+1)$ with identifier matching $r.id[\ell-1]$, hence r also has a publish link to it. Using the same arguments as in the proof of Lemma 3.1 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 $4^{-(\ell-i-1)} e^{-2i}$.

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

$$\sum_{i=0}^{\infty} \frac{1}{4^{\ell-1}} \left(4e^{-2}\right)^i = \frac{1}{4^{\ell-1}} \frac{1}{(1 - 4e^{-2})} \ .$$

Hence, the expected number of nodes among the $\Gamma 4^{\ell+4}$ nodes that match this criterion is bounded by

$$E[|u.P|] \leq \Gamma 4^{\ell+4} \frac{1}{4^{\ell-1}} \frac{1}{(1-4e^{-2})} = \frac{\alpha 4^5}{1-4e^{-2}} \ .$$

For any node u (regular or virtual), the probability that a node is a level-j direct link of u, i.e., that it matches the first j bits of v, is $1/2^j$. Hence, within $A_{j+d+2}(u)$, the expected number of such nodes is at most $\Gamma 2^{j+d+2}(1/2^j) = \Gamma 2^{d+2}$. By Lemma 3.1 above, the expected number of shadow router emulated by v is constant, each one of which has an expected constant number of direct links. Hence, the lemma follows. □

In is not hard to see that the expected in-degree of nodes is logarithmic by analogous arguments. Hence, we obtain the following statement:

Theorem 2 *The expected degree of all nodes is $O(M)$.*

4 From uniform density to growth-bounded metrics

In this section, we sketch how to extend the principles outlined above to more general metrics. The growth-bounded metrics we consider satisfies the following condition. There exists a constant parameter $\Delta > 1$ such that for every node v, and every radius $\rho \geq 1$, we have

$$|N(v, 2\rho)| \leq \Delta |N(v, \rho)| \ .$$

Growth-bounded metrics are considered in [8], and are slightly more general than the model assumed in [10, 11, 12], in which a bound is also placed on growth-density from below.

In order to adapt our mechanisms to this general network model, we need to change the network so as to maintain the following two properties: (i) First, for every k, we need that $A_k(v)$ will contain an expected constant number of routers with a specific length-k prefix. This property is required in order to maintain the expected number of links per router a constant. It is used in the proofs of Lemma 3.1 and Lemma 3.5. (ii) Second, for the constant stretch, we require that for every two nodes u, v, and every k, if $u \in A_k(v)$ then $A_k(v)$ is proportional to $A_{k+1}(u)$ and vice versa. This is required so that the k'th publishing step (w_k) and the k'th search step (x_k) will be in proportional distance from the target (t) and the source (s), respectively. The reader should verify that properties (i) and (ii) suffice to uphold (with slightly varying constants) the proofs of Lemma 3.3, Lemma 3.4 and Theorem 1.

For property (i), the technique we employ is to **re-define** the neighborhood $A_k(v)$ to be the smallest ball around v containing αB^k nodes, where B is a new radix we choose for identifiers, to be defined shortly, and α is chosen so that $B\alpha^{-1} < 1$. We continue to use the notation $a_k(v)$ for the radius of $A_k(v)$. For (ii), we simply choose $B \geq \Delta^2$ (accordingly, we set $M = \log_B N$). The next lemma shows that this indeed provides the neighborhood-containment behavior we desire, and also gives an explicit ratio bound between $a_k(v)$ and $a_{k+1}(v)$.

Lemma 4.1 *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 $A_i(x) = N(x, r)$. 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 αB^{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 $A_{i+1}(x) \supseteq N(x, B^{\log_\Delta 2} r)$. □

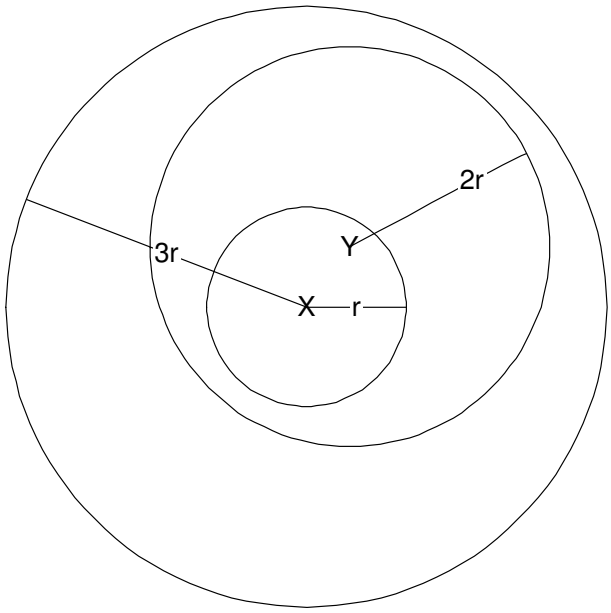

Figure 1: The circles $N(x, r)$, $N(y, 2r)$, and $N(x, 3r)$

5 From a static design to dynamic deployment

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

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, $R_1 \ldots R_{(n+1)}$, an identifier of M uniformly independent random radix B digits. Note that due to a significant change in the number of nodes, the parameter $M = \log_B N$ 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 et al. propose in [7] 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 guarantees.

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 3.

For a router $v.r$ with level ℓ, 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 algorithm for locating all of the required links for a router $v.r$ is done by routing from the node nearest to v to a node with a length-$(\ell+2)$ prefix matching r. From that node, by following existing incoming router links backward, suitable nodes within the required vicinity are guaranteed to be found.

Leave. Finally, when a regular node x of level ℓ 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 complexity of the join algorithm is $O(n^2)$ messages, and $O(M)$ nodes changing their state. Leave has $O(M)$ message complexity, and $O(M)$ state changes.

References

[1] I. Abraham, C. Gavoille, D. Malkhi, and N. Nisan. Stretch $(3+\varepsilon)$ name independent compact routing. Submitted for publication.

[2] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1+\varepsilon)$ locality aware networks for DHTs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, 2004.

[3] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka. Compact routing with name independence. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 184–192. ACM Press, 2003.

[4] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive routing. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 479–489. ACM Press, 1989.

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

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

[7] K. Hildrum, J. Kubiatowicz, and S. Rao. Another way to find the nearest neighbor in growth-restricted metrics. Technical Report UCB/CSD-03-1267, UC Berkeley, Computer Science Division, August 2003.

[8] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *ACM Symposium on Theory of Computing (STOC '02)*, 2002.

[9] 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)*, pages 82–89, October 2002.

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

[11] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[12] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 2003.