Overlay Networks for Peer-to-Peer Networks

Andréa W. Richa[*] Christian Scheideler[†]

November 17, 2005

1 Introduction

Every distributed system must be based on some kind of logical interconnection structure, also called an *overlay network* , that allows its sites to exchange information. Once distributed systems become large enough, one has to deal with sites continuously entering and leaving the system, simply because sites may fail and have to be replaced by new sites or because additional resources have to be added to preserve the functionality of the system. Hence, in general, a distributed system supporting any service has to have an overlay network supporting joining, leaving and routing between the sites, and without a scalable implementation of such a network, the field of scalable distributed systems does not really exist.

Scalability is especially critical for peer-to-peer systems . The basic idea of peer-to-peer systems is to have an open self-organizing system of peers that does not rely on any central server and where peers can join and leave at will. This has the benefit that individuals can cooperate without fees or an investment in additional high-performance hardware. Also, peer-to-peer systems can make use of the tremendous amount of resources (such as computation and storage) that otherwise sits idle on individual computers when they are not in use by their owners.

If we want a scalable peer-to-peer system, then joining, leaving and routing between the sites should be

[*]Department of Computer Science, Arizona State University, Tempe, AZ 85281, USA, `aricha@asu.edu`. This work was supported by NSF CAREER grant CCR-9985284.

[†]Institute for Computer Science, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany, `scheideler@in.tum.de`.

performed with at most polylogarithmic work in the size of the system. This implies that the maximum degree and the diameter of the overlay network should be at most polylogarithmic as well. The overlay network should also be well-connected so that it is robust against faulty peers. The well-connectedness of a graph is usually measured by its expansion, which we will formally define later in this chapter. Another important parameter is the stretch factor of an overlay network, which measures by how much the length of a shortest route between two nodes v and w in the overlay network is off from a shortest route from v to w when using the underlying physical network.

To summarize, we are seeking operations Join, Leave, and Route so that for any sequence of join, leave and route requests,

- the *work* of executing these requests is as small as possible,
- the *degree, diameter and stretch factor* of the resulting network are as small as possible, and
- the *expansion* of the resulting network is as large as possible.

Hence, we are dealing with multi-objective optimization problems for which we want to find good approximate solutions. To address these problems, we first introduce some basic notation and techniques for constructing overlay networks (Section 2). Afterwards, we start with supervised overlay network designs (i.e., the topology is maintained by a supervisor but routing is done on a peer-to-peer basis), and then we present various decentralized overlay network designs (i.e., the topology is maintained by the peers themselves). For simplicity, we assume that no peer fails and that the peers always execute the operations in a correct and timely manner (though in practice this might not be the case).

2 Basic notation and techniques

We start with some basic notation. A graph $G = (V, E)$ consists of a node set V and an edge set $E \subseteq V \times V$. We will only consider directed graphs. The *in-degree* of a node is the number of incoming edges, the *out-degree* of a node is the number of outgoing edges, and the *degree* of a node is the number of incoming and outgoing edges. Given two nodes v and w, let $d(v, w)$ denote the length of a shortest directed path from v

to w in G. G is strongly connected if $d(v,w)$ is finite for every pair $v,w \in V$. In this case,

$$D = \max_{v,w \in V} d(v,w)$$

is the *diameter* of G. The *expansion* of G is defined as

$$\alpha = \min_{S \subseteq V,\, |S| \leq |V|/2} \frac{|\Gamma(S)|}{|S|} .$$

where $\Gamma(S) = \{v \in V \setminus S \mid \exists u \in S : \ (u,v) \in E\}$ is the neighbor set of S. The following relationship between the expansion and diameter of a graph is easy to show.

Fact 1.1 *For any graph G with expansion α, the diameter of G is in $O(\alpha^{-1} \log n)$.*

The vast majority of overlay networks for peer-to-peer systems suggested in the literature is based on the concept of virtual space . That is, every site is associated with a point in some space U and connections between sites are established based on rules how to interconnect points in that space. In this case, the following operations need to be implemented:

- Join(p): add new peer p to the network by choosing a point in U for it
- Leave(p): remove peer p from the network
- Route(m, x): route message m to point x in U

Several virtual space approaches are known. The most influential techniques are the hierarchical decomposition technique, the continuous-discrete technique, and the prefix technique. We will give a general outline of each technique in this section. At the end of this section, we present two important families of graphs that we will use later in this chapter to construct dynamic overlay networks.

2.1 The hierarchical decomposition technique

Consider the space $U = [0,1]^d$ for some fixed $d \geq 1$. The *decomposition tree* $T(U)$ of U is an infinite binary tree in which the root represents U and for every node v representing a subcube U' in U, the children of v represent two subcubes U'' and U''', where U'' and U''' are the result of cutting U' in the middle at the smallest dimension in which U' has a maximum side length. The subcubes U'' and U''' are closed, i.e., their intersection gives the cut. Let every edge to a left child in $T(U)$ be labeled with 0 and every edge to a right child in $T(U)$ be labeled with 1. Then the label of a node v, $\ell(v)$, is the sequence of all edge labels encountered when moving along the unique path from the root of $T(U)$ downwards to v. For $d = 2$, the result of this decomposition is shown in Figure 1.1.

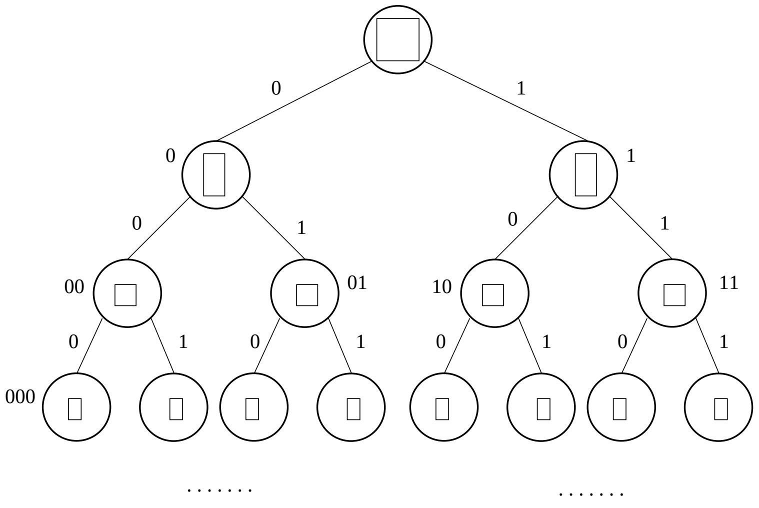

Figure 1.1: The decomposition tree for $d = 2$.

The goal is to map the peers to nodes in $T(U)$ so that the following conditions are met:

Condition 1.1

(1) The interiors of the subcubes associated with the (nodes assigned to the) peers are disjoint,

(2) the union of the subcubes of the peers gives the entire set U, and

(3) every peer p with subcube U_p is connected to all peers p' with subcubes $U_{p'}$ that are adjacent to U_p (i.e., $U_p \cap U_{p'}$ is a $d-1$-dimensional subcube).

In the 2-dimensional case, for example, condition (3) means that p and p' share a part of the cut line through their first common ancestor in $T(U)$. It is not difficult to see that the following result is true.

Fact 1.2 *Consider the space $U = [0,1]^d$ for some fixed d and suppose we have n peers. If the peers are associated with nodes that are within k levels of $T(U)$ and Condition 1.1 is satisfied, then the maximum degree of a peer is at most $(2d) \cdot 2^{k-1}$ and the diameter of the graph is at most $d \cdot n^{1/d} + 2(k-1)$.*

The diameter of the graph can be as large as $d \cdot n^{1/d}$ and therefore too large for a scalable graph if d is fixed, but its degree is small as long as $k = O(\log \log n)$.

An example of a peer-to-peer system using the hierarchical decomposition technique is CAN [20]. In the original CAN construction, a small degree is achieved by giving each peer p a label $\ell(p)$ consisting of a (sufficiently long) random bit string when it joins the system. This bit string is used to route p to the unique peer p' that is reached when traversing the tree $T(U)$ according to $\ell(p)$ (a 0 bit means "go left" and a 1 bit means "go right") until a node v is reached that is associated with a peer. (Such a node must always exist if there is at least one peer in the system and the two rules of assigning peers to nodes in Condition 1.1 are satisfied.) One of p and p' is then placed in the left child of v and the other in the right child of v. Leave operations basically reverse join operations so that Condition 1.1 is maintained. Due to the use of a random bit sequence, one can show that the number of levels the peers are apart is indeed $O(\log \log n)$, as desired, but it can also be as bad as that. Strategies that achieve a more even level balancing were subsequently proposed in several papers (see, e.g., [23] and the references therein).

2.2 The continuous-discrete technique

The basic idea underlying the continuous-discrete approach [16] is to define a continuous model of graphs and to apply this continuous model to the discrete setting of a finite set of peers. A well-known peer-to-peer system that uses an approach closely related to the continuous-discrete approach is Chord [22].

Consider the d-dimensional space $U = [0,1)^d$, and suppose that we have a set F of continuous functions $f_i : U \to U$. Then we define E_F as the set of all pairs $(x,y) \in U^2$ with $y = f_i(x)$ for some i. Given any subset $S \subseteq U$, let $\Gamma(S) = \{y \in U \setminus S \mid \exists x \in S : (x,y) \in E_F\}$. If $\Gamma(S) \neq \emptyset$ for every $S \subset U$, then F is said to be *mixing*. If F does not mix, then there are disconnected areas in U.

Consider now any set of peers V, and let $S(v)$ be the subset in U that has been assigned to peer v. Then the following conditions have to be met:

Condition 1.2

(1) $\cup_v S(v) = U$ *and*

(2) for every pair of peers v and w it holds that v is connected to w if and only if there are two points $x, y \in U$ with $x \in S(v)$, $y \in S(w)$ and $(x, y) \in E_F$.

Let $G_F(V)$ be the graph resulting from the conditions above. Then the following fact is easy to see.

Fact 1.3 *If F is mixing and $\cup_v S(v) = U$, then $G_F(V)$ is strongly connected.*

To bound the diameter of $G_F(V)$, we introduce some further notation. For any point x and any $\epsilon \in [0, 1)$, let $B(x, \epsilon)$ denote the d-dimensional ball of volume ϵ centered at x. For any two points x and y in U let $d_n(x, y)$ denote the shortest sequence $(s_1 s_2 s_3 \ldots s_k) \in \mathbb{N}^k$ so that there are two points $x' \in B(x, 1/n)$ and $y' \in B(y, 1/n)$ with $f_{s_1} \circ f_{s_2} \circ \ldots f_{s_k}(x') = y'$. Then we define the diameter of F as

$$D(n) = \max_{x,y \in U} d_n(x, y)$$

Using this definition, it holds:

Fact 1.4 *If $\cup_v S(v) = U$ and every $S(v)$ contains a ball of volume at least $1/n$ then $G_F(V)$ has a diameter of at most $D(n)$.*

Also the expansion of $G_F(V)$ can be bounded with a suitable parameter for F, but it is easier to consider explicit examples here, and therefore we defer a further discussion to Section 4.1.

2.3 The prefix technique

The prefix technique was first presented in [18, 17] and first used in the peer-to-peer world by Pastry [7] and Tapestry [24]. Given a label $\ell = (\ell_1 \ell_2 \ell_3 \ldots)$, let $\text{prefix}_i(\ell) = (\ell_1 \ell_2 \ldots \ell_i)$ for all $i \geq 1$ and $\text{prefix}_0(\ell) = \epsilon$, the empty label.

In the prefix technique, every peer node v is associated with a unique label $\ell(v) = \ell(v)_1 \ldots \ell(v)_k$, where each $\ell(v)_i \in \{0, \ldots, b-1\}$, for some constant $b \geq 2$ and sufficiently large k and the following condition has to be met concerning connections between the nodes.

Condition 1.3 *For every peer v, every digit $\alpha \in \{0, \ldots, b-1\}$, and every $i \geq 0$, v has a link to a peer node w with $\text{prefix}_i(\ell(v)) = \text{prefix}_i(\ell(w))$ and $\ell(w)_{i+1} = \alpha$, if such a node w exists.*

Since for some values of i and α there can be many nodes w satisfying the condition above, a rule has to be specified which of these nodes w to pick. For example, a peer may connect to the geographically closest peer w, or a peer may connect to a peer w it has the best connection to. The following result is easy to show:

Fact 1.5 *If the maximum length of a node label is L and the node labels are unique, then any rule of choosing a node w as in Condition 1.3 guarantees strong connectivity. Moreover, the maximum out-degree of a node and the diameter of the network are at most L.*

However, the in-degree, i.e., the number of incoming connections, can be quite high, depending on the rule. An easy strategy guaranteeing polylogarithmic in- and out-degree and logarithmic diameter is that every node chooses a random binary sequence as its label, and a node v connects to the node w among the eligible candidates with the closest distance to v, i.e., $|\ell(v) - \ell(w)|$ is minimized. This rule also achieves a good expansion but not a good stretch factor. In order to address the stretch factor, other rules are necessary that are discussed in Section 4.2.

2.4 Basic classes of graphs

We will apply our basic techniques above to two important classes of graphs: the hypercube and the de Bruijn graph. They are defined as follows.

Definition 1.1 *For any $d \in \mathbb{N}$, the d-*dimensional hypercube *is an undirected graph $G = (V, E)$ with $V = \{0,1\}^d$ and $E = \{\{v, w\} \mid H(v, w) = 1\}$ where $H(v, w)$ is the Hamming distance between v and w.*

Definition 1.2 *For any $d \in \mathbb{N}$, the d-*dimensional de Bruijn graph *is an undirected graph $G = (V, E)$ with node set $V = \{v \in \{0,1\}^d\}$ and edge set E that contains all edges $\{v, w\}$ with the property that $w \in \{(x, v_{d-1}, \ldots, v_1) : x \in \{0,1\}\}$, where $v = (v_{d-1}, \ldots, v_0)$.*

3 Supervised overlay networks

A *supervised overlay network* is a network formed by a supervisor but in which all other activities can be performed on a peer-to-peer basis without involving the supervisor. It can therefore be seen as being between server-based overlay networks and pure peer-to-peer overlay networks. In order for a supervised network to be highly scalable, two central requirements have to be satisfied:

1. The supervisor needs to store at most a polylogarithmic amount of information about the system at any time (e.g., if there are n peers in the system, storing contact information about $O(\log^2 n)$ of these peers would be fine), and
2. the supervisor needs at most a constant number of messages to include a new peer into or exclude an old peer from the network.

The second condition makes sure that the work of the supervisor to include or exclude peers from the system is kept at a minimum. First, we present a general strategy of constructing supervised overlay networks, which combines the hierarchical decomposition technique with the continuous-discrete technique and the recursive labeling technique below, and then we give some explicit examples that achieve near-optimal results for the cost of the join, leave and route operations as well as the degree, diameter and expansion of the network.

3.1 The recursive labeling technique

In the recursive labeling approach, the supervisor assigns a *label* to every peer that wants to join the system. The labels are represented as binary strings and are generated in the following order:

$$0, 1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, 0111, 1001, 1011, \ldots$$

Basically, when stripping off the least significant bit, then the supervisor is first creating all binary numbers of length 0, then length 1, then length 2, and so on. More formally, consider the mapping $\ell : \mathbb{N}_0 \to \{0,1\}^*$ with the property that for every $x \in \mathbb{N}_0$ with binary representation $(x_d \ldots x_0)_2$ (where d is minimum possible),

$$\ell(x) = (x_{d-1} \ldots x_0 x_d) \ .$$

Then ℓ generates the sequence of labels displayed above. In the following, it will also be helpful to view labels as real numbers in $[0, 1)$. Let the function $r : \{0,1\}^* \to [0,1)$ be defined so that for every label $\ell = (\ell_1 \ell_2 \ldots \ell_d) \in \{0,1\}^*$, $r(\ell) = \sum_{i=1}^{d} \frac{\ell_i}{2^i}$. Then the sequence of labels above translates into

$$0,\ 1/2,\ 1/4,\ 3/4,\ 1/8,\ 3/8,\ 5/8,\ 7/8,\ 1/16,\ 3/16,\ 5/16,\ 7/16,\ 9/16,\ \ldots$$

Thus, the more labels are used, the more densely the $[0, 1)$ interval will be populated. When using the recursive approach, the supervisor aims to maintain the following condition at any time:

Condition 1.4 *The set of labels used by the peers is* $\{\ell(0), \ell(1), \dots, \ell(n-1)\}$*, where* n *is the current number of peers in the system.*

This condition is preserved when using the following simple strategy:

- Whenever a new peer v joins the system and the current number of peers is n, the supervisor assigns the label $\ell(n)$ to v and increases n by 1.
- Whenever a peer w with label ℓ wants to leave the system, the supervisor asks the peer with currently highest label $\ell(n-1)$ to take over the role of w (and thereby change its label to ℓ) and reduces n by 1.

3.2 Putting the pieces together

We assume that we have a single supervisor for maintaining the overlay network. In the following, the label assigned to some peer v will be denoted as ℓ_v. Given n peers with unique labels, we define the *predecessor* $\text{pred}(v)$ of peer v as the peer w for which $r(\ell_w)$ is closest from below to $r(\ell_v)$, and we define the *successor* $\text{succ}(v)$ of peer v as the peer w for which $r(\ell_w)$ is closest from above to $r(\ell_v)$ (viewing $[0, 1)$ as a ring in both cases). Given two peers v and w, we define their *distance* as

$$\delta(v, w) = \min\{(1 + r(\ell_v) - r(\ell_w)) \bmod 1,\ (1 + r(\ell_w) - r(\ell_v)) \bmod 1\} .$$

In order to maintain a doubly linked cycle among the peers, we simply have to maintain the following condition:

Condition 1.5 *Every peer* v *in the system is connected to* $\text{pred}(v)$ *and* $\text{succ}(v)$*.*

Now, suppose that the labels of the peers are generated via the recursive strategy above. Then we have the following properties:

Lemma 1.1 *Let* n *be the current number of peers in the system, and let* $\bar{n} = 2^{\lfloor \log n \rfloor}$*. Then for every peer* $v \in V$*,* $|\ell_v| \le \lceil \log n \rceil$ *and* $\delta(v, \text{pred}(v)) \in \{1/(2\bar{n}), 1/\bar{n}\}$*.*

So the peers are approximately evenly distributed in $[0, 1)$ and the number of bits for storing a label is almost as low as it can be without violating the uniqueness requirement.

Now, recall the hierarchical decomposition approach. The supervisor will assign every peer p to the unique node v in $T(U)$ at level $\log(1/\delta(p, \mathrm{pred}(p)))$ with ℓ_v being equal to ℓ_p (padded with 0's to the right so that $|\ell_v| = |\ell_p|$). As an example, if we have 4 peers currently in the system, then the mapping of peer labels to node labels is

$$0 \rightarrow 00,\ 1 \rightarrow 10,\ 01 \rightarrow 01,\ 11 \rightarrow 11$$

With this strategy, it follows from Lemma 1.1 that Fact 1.2 applies with $k = 2$.

Consider now any family F of functions acting on some space $U = [0, 1)^d$ and let $C(p)$ be the subcube of the node in $T(U)$ that p has been assigned to. Then the goal of the supervisor is to maintain the following condition at any time.

Condition 1.6 *For the current set V of peers in the system it holds that*

1. *the set of labels used by the peers is $\{\ell(0), \ell(1), \ldots, \ell(n-1)\}$, where $n = |V|$,*
2. *every peer v in the system is connected to* $\mathrm{pred}(v)$ *and* $\mathrm{succ}(v)$*, and*
3. *there is an edge (v, w) for every pair of peers v and w for which there is an edge $(x, y) \in E_F$ with $x \in C(v)$ and $y \in C(w)$.*

3.3 Maintaining Condition 1.6

Next we describe the actions that the supervisor has to perform in order to maintain Condition 1.6 during a join or leave operation. We start with the following important fact.

Fact 1.6 *Whenever a new peer v enters the system, then* $\mathrm{pred}(v)$ *has all the connectivity information v needs to satisfy Condition 1.6(3), and whenever an old peer w leaves the system, then it suffices that it transfers all of its connectivity information to* $\mathrm{pred}(w)$ *in order to maintain Condition 1.6(3).*

The first part of the fact follows from the observation that when v enters the system, then the subcube of $\mathrm{pred}(v)$ splits into two subcubes where one resides at $\mathrm{pred}(v)$ and the other is taken over by v. Hence, if

$\mathrm{pred}(v)$ passes all of its connectivity information to v, then v can establish all edges relevant for it according to the continuous-discrete approach. The second part of the fact follows from the observation that the departure of a peer is the reverse of the insertion of a peer.

Thus, if the peers take care of the connections in Condition 1.6(3), the only part that the supervisor has to take care of is maintaining the cycle. For this we require the following condition.

Condition 1.7 *At any time, the supervisor stores the contact information of* $\mathrm{pred}(v)$*,* v*,* $\mathrm{succ}(v)$*, and* $\mathrm{succ}(\mathrm{succ}(v))$ *where* v *is the peer with label* $\ell(n-1)$*.*

In order to satisfy Condition 1.7, the supervisor performs the following actions. If a new peer w joins, then the supervisor

- informs w that $\ell(n)$ is its label, $\mathrm{succ}(v)$ is its predecessor, and $\mathrm{succ}(\mathrm{succ}(v))$ is its successor,
- informs $\mathrm{succ}(v)$ that w is its new successor,
- informs $\mathrm{succ}(\mathrm{succ}(v))$ that w is its new predecessor,
- asks $\mathrm{succ}(\mathrm{succ}(v))$ to send its successor information to the supervisor, and
- sets $n = n + 1$.

If an old node w leaves and reports ℓ_w, $\mathrm{pred}(w)$, and $\mathrm{succ}(w)$ to the supervisor (recall that we are assuming graceful departures), then the supervisor

- informs v (the node with label $\ell(n-1)$) that ℓ_w is its new label, $\mathrm{pred}(w)$ is its new predecessor, and $\mathrm{succ}(w)$ is its new successor,
- informs $\mathrm{pred}(w)$ that its new successor is v and $\mathrm{succ}(w)$ that its new predecessor is v,
- informs $\mathrm{pred}(v)$ that $\mathrm{succ}(v)$ is its new successor and $\mathrm{succ}(v)$ that $\mathrm{pred}(v)$ is its new predecessor,
- asks $\mathrm{pred}(v)$ to send its predecessor information to the supervisor and to ask $\mathrm{pred}(\mathrm{pred}(v))$ to send its predecessor information to the supervisor, and
- sets $n = n - 1$.

Thus, the supervisor only needs to handle a small constant number of messages for each arrival or departure of a peer, as desired. Next we look at two examples resulting in scalable supervised overlay networks.

3.4 Examples

For a supervised hypercubic network, simply select F as the family of functions on $[0, 1)$ with $f_i(x) = x + 1/2^i \pmod 1$ for every $i \geq 1$. Using our framework, this gives an overlay network with degree $O(\log n)$, diameter $O(\log n)$, and expansion $O(1/\sqrt{\log n})$, which matches the properties of ordinary hypercubes.

For a supervised de Bruijn network, simply select F as the family of functions on $[0, 1)$ with $f_0(x) = x/2$ and $f_1(x) = (1 + x)/2$. Using our framework, this gives an overlay network with degree $O(1)$, diameter $O(\log n)$, and expansion $O(1/\log n)$, which matches the properties of ordinary de Bruijn graphs.

In both networks, routing with logarithmic work can be achieved by using the bit adjustment strategy.

4 Decentralized overlay networks

Next we show that scalable overlay networks can also be maintained without involving a supervisor. We only discuss examples for the latter two basic techniques in Section 2 since the hierarchical decomposition technique cannot yield networks of polylogarithmic diameter.

4.1 Overlay networks based on the continuous-discrete approach

Similar to the supervised approach, we first show how to maintain a hypercubic overlay network, and then we show how to maintain a de Bruijn-based overlay network.

Maintaining a dynamic hypercube

Let $U = [0, 1)$ and consider the family F of functions on $[0, 1)$ with $f_i(x) = x + 1/2^i \pmod 1$ for every $i \geq 1$. Given a set of points $V \subset [0, 1)$, we define the region $S(v)$ associated with point v as the interval $(\text{pred}(v), v)$ where $\text{pred}(v)$ is the closest predecessor of v in V and U is seen as a ring. The following result follows from [4]:

Theorem 1.1 *If every peer is given a random point in $[0,1)$, then the graph $G_F(V)$ with $|V| = n$ resulting from the continuous-discrete approach has a degree of $O(\log^2 n)$, a diameter of $O(\log n)$ and an expansion of $\Omega(1/\log n)$, with high probability.*

Suppose that Condition 1.2 is satisfied for our family of hypercubic functions. Then it is fairly easy to route a message from any point $x \in [0,1)$ to any point $y \in [0,1)$ along edges in $G_F(V)$:

Consider the path P in the continuous space that results from using a bit adjustment strategy in order to get from x to y. That is, given that $x_1x_2x_3\ldots$ is the bit sequence for x and $y_1y_2y_3\ldots$ is the bit sequence for y, P is the sequence of points $z_0 = x_1x_2x_3\ldots, z_1 = y_1x_2x_3\ldots, z_2 = y_1y_2x_3\ldots, \ldots, y_1y_2y_3\ldots = y$. Of course, P may have an infinite length, but simulating P in $G_F(V)$ only requires traversing a finite sequence of edges:

We start with the region $S(v)$ containing $x = z_0$. Then we move along the edge (v,w) in $G_F(V)$ to the region $S(w)$ containing z_1. This edge must exist because we assume that Condition 1.2 is satisfied. Then we move along the edge (w,w') simulating (z_1,z_2), and so on, until we reach the node whose region contains y. Using this strategy, it holds:

Theorem 1.2 *Given a random node set $V \subset [0,1)$ with $|V| = n$, it takes at most $O(\log n)$ hops, with high probability, to route in $G_F(V)$ from any node $v \in V$ to any node $w \in V$.*

Next we explain how nodes can join and leave. Suppose that a new node v contacts some node already in the system to join the system. Then v's request is first sent to the node u in V with $u = \text{succ}(v)$, which only takes $O(\log n)$ hops according to Theorem 1.2. u forwards information about all of its incoming and outgoing edges to v, deletes all edges that it does not need any more, and informs the corresponding endpoints about this. Because $S(v) \subseteq S(u)$ for the old $S(u)$, the edges reported to v are a superset of the edges that it needs to establish. v checks which of the edges are relevant for it, informs the other endpoint for each relevant edge, and removes the others.

If a node v wants to leave the network, it simply forwards all of its incoming and outgoing edges to $\text{succ}(v)$. $\text{succ}(v)$ will then merge these edges with its existing edges and notifies the endpoints of these edges about the changes.

Combining Theorem 1.1 and Theorem 1.2 we obtain:

Theorem 1.3 *It takes a routing effort of $O(\log n)$ hops and an update work of $O(\log^2 n)$ messages that can be processed in $O(\log n)$ communication rounds in order to execute a join or leave operation.*

Maintaining a dynamic deBruijn graph

Next we show how to dynamically maintain a deBruijn graph. Let $U = [0,1)$ and F consist of two functions, f_0 and f_1, where $f_i(x) = (i+x)/2$ for each $i \in \{0,1\}$. Then one can show the following result:

Theorem 1.4 *If the peers are mapped to random points in $[0,1)$, then the graph $G_F(V)$ resulting from the continuous-discrete approach has a degree of $O(\log n)$, diameter of $O(\log n)$ and node expansion of $\Omega(1/\log n)$, with high probability.*

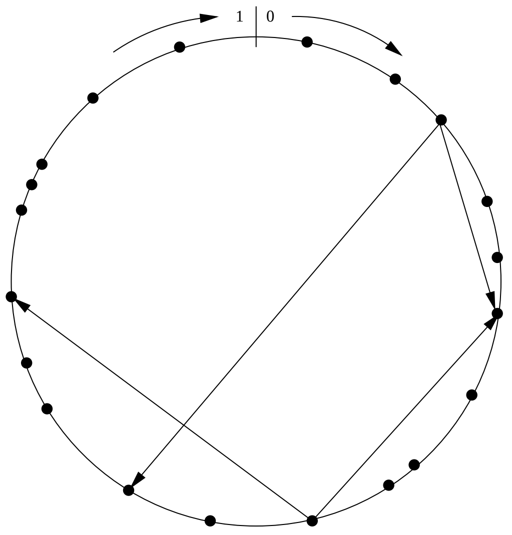

Figure 1.2: An example of a dynamic de Bruijn network (only some short-cut pointers for two nodes are given).

Next we show how to route in the de Bruijn network. Suppose that Condition 1.2 is satisfied. Then we use the following trick to route a message from any point $x \in [0,1)$ to any point $y \in [0,1)$ along edges in $G_F(V)$.

Let z be a randomly chosen point in $[0,1)$. Let $x_1x_2x_3\ldots$ be the binary representation of x, $y_1y_2y_3\ldots$ be the binary representation of y, and $z_1z_2z_3$ be the binary representation of z. Let P be the path along the points $x = x_1x_2x_3\ldots, z_1x_1x_2\ldots, z_2z_1x_1\ldots, \ldots$ and P' be the path along the points $y = y_1y_2y_3\ldots,$

$z_1y_1y_2\ldots, z_2z_1y_1\ldots, \ldots$ Then we simulate moving along the points in P by moving along the corresponding edges in $G_F(V)$ until we hit a node $w \in V$ with the property that $S(w)$ contains a point in P and a point in P'. At that point, we follow the points in P' backwards until we arrive at the node $w' \in W$ that contains y in $S(w')$. Using this strategy, it holds:

Theorem 1.5 *Given a random node set $V \subset [0,1)$ with $|V| = n$, it takes at most $O(\log n)$ hops, with high probability, to route in $G_F(V)$ from any point $x \in [0,1)$ to any point $y \in [0,1)$.*

Joining and leaving the network is done in basically the same way as in the hypercube, giving the following result:

Theorem 1.6 *It takes a routing effort of $O(\log n)$ hops and an update work of $O(\log n)$ messages that can be processed in a logarithmic number of communication rounds in order to execute a join or leave operation in the dynamic de Bruijn graph.*

4.2 Overlay networks based on prefix connections

The efficiency of routing on an overlay network is quite often measured in terms of the number of hops (neighbor links) followed by a message. While this measure indicates the latency of a message in the overlay network, it fails to convey the complexity of the given operation with respect to the original underlying network. In other words, while in the overlay network all overlay links may have the same cost, this is not true when those overlay links are translated back into paths in the underlying network. Hence, if one is to analyze the routing performance of an overlay network in a way that is more meaningful for real-life scenarios, one needs to take into account the different internode communication costs in the underlying network when charging for the cost of following a path in the overlay network. For example, a hop from a peer in the USA to a peer in Europe costs a lot more (in terms of reliability, speed, cost of deploying and maintaining the link, etc.) than a hop going between two peers in a local area network. In brief, keeping routing local is important: It may make sense to route a message originated in Phoenix for a destination peer in San Francisco through Los Angeles, but not through a peer in Europe.

In this section, we present peer-to-peer overlay network design schemes which take locality into account and which are able to achieve constant stretch factors, while keeping polylogarithmic degree and diameter, and polylogarithmic complexity for join and leave operations. All of the work in this section assumes that the underlying peer-to-peer system is a growth-bounded network, which we define a few paragraphs later.

Peers communicate with one another by means of messages; each message consists of at least one word. We assume that the underlying network supports reliable communication. We define the cost of communication by a function $c : V^2 \rightarrow \Re$. This function c is assumed to reflect the combined effect of the relevant network parameter values, such as latency, throughput, congestion, etc. In other words, for any two peers u and v in V, $c(u, v)$ is the cost of transmitting a single-word message from u to v. We assume that c is symmetric and satisfies the triangle inequality. The cost of transmitting a message of length l from peer u to peer v is given by $f(l)c(u, v)$, where $f : \mathbf{N} \rightarrow \Re^+$ is any nondecreasing function such that $f(1) = 1$.

A *growth-bounded* network satisfies the following property: Given any u in V and any real r, let $B(u, r)$ denote the set of peers v such that $c(u, v) \leq r$. We refer to $B(u, r)$ as the *ball* of *radius* r around u. We assume that there exist a real constant Δ such that for any peer u in V and any real $r \geq 1$, we have

$$|B(u, 2r)| \quad \leq \quad \Delta|B(u, r)|. \tag{1.1}$$

In other words, the number of peers within radius r from u grows polynomially with r. This network model has been validated by both theoreticians and practitioners as to model well existing internetworking topologies [24, 7, 2, 13].

Plaxton, Rajaraman and Richa (PRR) in [18, 17] pioneered the work on locality-aware routing schemes in dynamic environments. Their work actually addresses a more general problem — namely the *object location problem* — than that of designing efficient overlay networks with respect to the parameters outlined in Section 1. In that early work, Plaxton, et al. formalize the problem of object location in a peer-to-peer environment, pinpointing the issue of locality and developing a formal framework under which object location schemes have been rigorously analyzed. In the object location problem, peers seek to find objects in a dynamic and fully distributed environment, where multiple (identical) copies of an object may exist in the network: The main goal is to be able to locate and find a copy of an object within cost that is proportional to the cost of retrieving the closest copy of the object to the requesting peer, while being able to efficiently support these operations in a dynamic peer-to-peer environment where copies of the objects are continuously inserted and removed from the network.

Our overlay network design problem can be viewed as a subset of the object location problem addressed by PRR, where each object is a peer (and hence there exists a single copy of each object in the network). Hence the results in the PRR scheme and other object location schemes to follow, in particular the LAND scheme to be addressed later in this section, directly apply to our overlay network design problem. Both PRR and LAND assume a growth-bounded network model. For these networks, the LAND scheme provides the best currently known overlay design scheme with $1+\epsilon$ stretch, and polylogarithmic bounds on diameter and degree, for any fixed constant $\epsilon > 0$. Combining the LAND scheme with a technique by Hildrum et al [12] to find nearest neighbors enable us to also attain polylogarithmic work for JOIN and LEAVE operations. Since the LAND scheme heavily relies on the PRR scheme, we will present the latter in more detail and then highlight the changes introduced by the LAND scheme. We will address both schemes under the light of overlay routing, rather than the object location problem originally addressed by these schemes.

In the PRR and related schemes, each peer p will have two attributes in addition to its exact location in the network (which is unknown to other peers): a virtual location x in U, and a *label* $\ell(x)$ *generated independently and uniformly at random.* We call the virtual location x of p the *peer identifier* x, or simply, the *ID* x. In the remainder of this section, we will *indistinctly use the ID* x *to denote the peer* p *itself.*

4.2.1 The PRR scheme

The original PRR scheme assumes that we have a growth-bounded network with the extra assumption of also having a lower bound on the rate of growth. Later work that evolved from the PRR scheme (e.g., the LAND scheme) showed that this assumption could be dropped by slightly modifying the PRR scheme. The PRR scheme and the vast majority of provably efficient object location schemes rely on a basic yet powerful technique called *prefix routing*, which we outlined in Section 4.2. Below, we re-visit prefix routing in the context of the PRR scheme.

Theorem 1.7 *The PRR scheme, when combined with a technique by Hildrum et al. for finding nearest neighbors, achieves an overlay peer-to-peer network with the following properties: expected constant stretch for* ROUTE *operations,* $O(\log n)$ *diameter, and, with high probability,* $O(\log^2 n)$ *degree and work for* JOIN, LEAVE *and* ROUTE *operations.*

Prefix Routing

The basic idea behind prefix routing (see also Section 4.2) is that the path followed in a routing operation will be guided solely by the ID y we are seeking for: Every time a ROUTE(m, y) request is forwarded from a peer u to a peer v in the overlay path, the prefix of $\ell(v)$ that (maximally) matches a prefix of the ID y is strictly larger than that of $\ell(u)$.

We now sketch the PRR prefix routing scheme (for more details, see [17]). Each peer x in the network is assigned a $(\log_b n)$-digit label[1] $\ell(x) = \ell(x)_1 \ldots \ell(x)_{\log_b n}$, where each $\ell(x)_i \in \{0, \ldots, b-1\}$, uniformly at random, for some large enough constant $b \geq 2$. Recall that each peer also has a unique $(\log_b n)$-digit ID which is independent of this label. We denote the ID of peer x by $x_1 \ldots x_{\log_b n}$, where each $x_i \in \{0, \ldots, b-1\}$.

The random labels are used to construct a *neighbor table* at each peer. For a base b sequence of digits $\gamma = \gamma_1 \ldots \gamma_m$, $\text{prefix}_i(\gamma)$ denotes the first i digits, $\gamma_1 \ldots \gamma_i$, of γ. For each peer x, each integer i between 1 and $\log_b n$, and each digit α between 0 and $b-1$, the neighbor table at peer x stores the $(\log_b n)$-digit ID of the *closest* peer y to x — i.e., the peer with minimum $c(x, y)$ — such that $\text{prefix}_{i-1}(\ell(x)) = \text{prefix}_{i-1}(\ell(y))$ and $\ell(y)_i = \alpha$. We call y the (i, α)-*neighbor* of peer x. There exists an edge between any pair of neighbor peers in the overlay network. Figure 1.3 illustrates the neighbor table of peer x for $b = 2$.

The degree of a peer in the overlay network is given by the size (number of entries) of its neighbor table. The size of the neighbor table at each peer, as constructed above, is $O(\log n)$. In the final PRR scheme (and other follow-up schemes) the neighbor table at a peer will have size polylogarithmic on n (namely $O(\log^2 n)$ in the PRR scheme), since a set of "auxiliary" neighbors at each peer will need to be maintained in order for the scheme to function efficiently. Each peer x will also need to maintain a set of "pointers" to the location of the subset of peers that were published at x by JOIN operations. In the case of routing, each peer will maintain $O(\log^2 n)$ such pointers with high probability [2]. We describe those pointers in more detail while addressing the JOIN operation in PRR.

The sequence of peers visited in the overlay network during a routing operation for ID y initiated by peer x will consist of the sequence $x = x^0, x^1, \ldots, x^q$, where x^i is the (i, y_i)-neighbor of peer x^{i-1}, for $1 \leq i \leq q$, and x^p is a peer that holds a pointer to y in the network ($p \leq \log_b n$). We call this sequence the *neighbor*

[1]Without loss of generality, assume that n is a power of b.

[2]With probability at least $1 - 1/p(n)$, where $p(n)$ is a polynomial function on n.

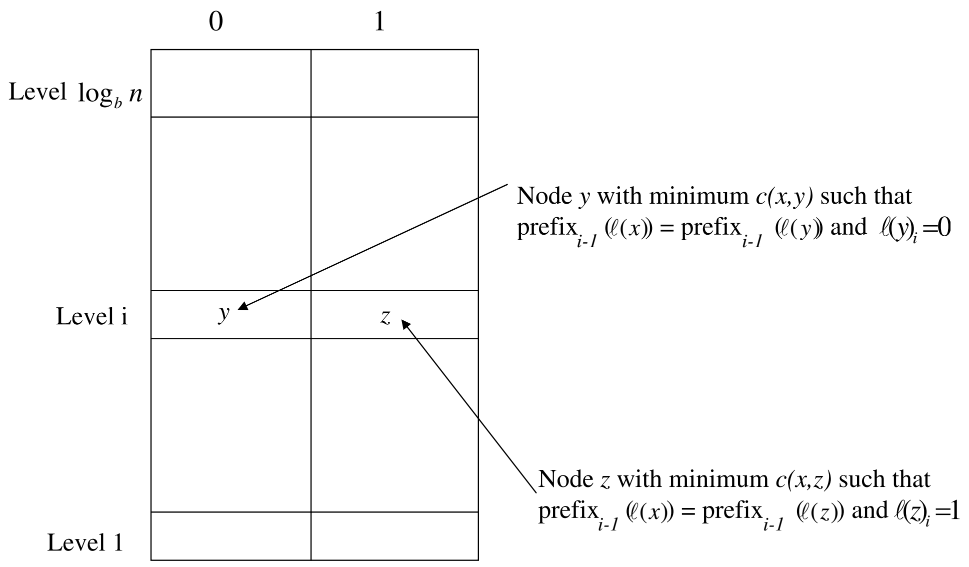
0
1
Level $\log_b n$
Node y with minimum $c(x,y)$ such that prefix$_{i-1}(\ell(x))$ = prefix$_{i-1}(\ell(y))$ and $\ell(y)_i = 0$
Level i
y
z
Node z with minimum $c(x,z)$ such that prefix$_{i-1}(\ell(x))$ = prefix$_{i-1}(\ell(z))$ and $\ell(z)_i = 1$
Level 1

Figure 1.3: The neighbor table of peer node x, for $b = 2$.

sequence of peer x for (peer ID) y. Below we explain in more detail the ROUTE, JOIN and LEAVE operations according to PRR.

Route, Join, and Leave

Before we describe a routing operation in the PRR scheme, we need to understand how JOIN and LEAVE operations are processed. Since we are interested in keeping low stretch, the implementation of such operations will be different in the PRR scheme than in the two other overlay network design techniques presented in this chapter. In a ROUTE(m, x) operation, we will, as in the hierarchical decomposition and continuous-discrete approaches, start by routing towards the virtual location x; however as we route toward this virtual location, as soon as we find some information on the network regarding the actual location of the peer p corresponding to x, we will redirect our route operation in order to reach p. This way, we will be able to show that the total stretch of a route operation is low (If we were to route all the way to the virtual location x in order to find information about the actual location of p, the total incurred stretch might be too large.).

There are two main components in a JOIN(p) operation: First, information about the location of peer p joining the peer-to-peer system needs to be published at the network so that subsequent ROUTE operations can indeed locate peer p; second, peer p needs to build its own neighbor table, and other peers in the network may need to update their neighbor tables given the presence of peer x in the network. Similarly, there are two main components in a LEAVE(p) operation: unpublishing any information about p's location in the network, and removing any entries containing peer p in the neighbor tables of other peers. We will address these two issues separately. For the moment, we will only be concerned with how information about p is published or unpublished in the network, since this is what we need in order to guarantee the success of a ROUTE operation. We will assume that the respective routing table entries are updated correctly upon the addition or removal of p from the system. Later, we will explain how the neighbor tables can be efficiently updated.

Whenever a peer p with ID x decides to join the peer-to-peer system, we place (*publish*) a pointer leading to the actual location of p in the network, which we call an *x-pointer*, at up to $\log_b n$ peers of the network. For convenience of notation, in the remainder of this section, we will *always use x to indistinctly denote both the ID of peer p and the peer p itself.* Let $x^0 = x, x^1, \ldots, x^{\log_b n - 1}$ be the neighbor sequence of peer x for node ID x. We place an x-pointer to x^{i-1} at *each* peer x^i in this neighbor sequence. Thus, whenever we find a peer with an x-pointer (at a peer x^j, $1 \le j \le \log_b n$) during a ROUTE(m, x) operation, we can forward the message all the way "down" the reverse neighbor sequence $x^j, \ldots, x^0 = x$ to the actual location of peer x.

The LEAVE(p) operation is the reverse of a JOIN operation: We simply *unpublish* (remove) all the x-pointers from the peers $x^0, \ldots, x^{\log_b n}$.

There is an implicit search tree associated with each peer ID y in prefix routing. Assume for a moment that there is only one peer r matching a prefix of the ID y in the largest number of digits in the network. At the end of this section, we address the case when this assumption does not hold. Let the search tree $T(y)$ for peer y be defined by the network edges in the neighbor sequences for peer ID y for each peer x in the network, where, for each edge of the type (x^{i-1}, x^i) in the neighbor sequence of x for y, we view x^i as the parent of x^{i-1} in the tree. The above implementation of the publish and unpublish operations trivially maintain the following invariant. Let $T_x(y)$ be the subtree rooted at x in $T(y)$.

Invariant 1.1 *If peer y belongs to $T_x(y)$, then peer x has a y-pointer.*

We now describe how a Route(m, y) operation initiated at peer x proceeds. Let $x^0 = x, x^1, \ldots, x^{\log_b n-1}$ be the neighbor sequence of peer x for y. Starting with $i = 0$, peer x^i first checks whether it has a y-pointer. If it does then x^i will forward the message m using its y-pointer down the neighbor sequence that was used when publishing information about peer y during a Join(y) operation. More specifically, let j be the maximum index such that $\text{prefix}_j(\ell(x^i)) = \text{prefix}_j(y)$ (note that $j \geq i$). Then x^i must be equal to y^j, where $y = y^0, y^1, \ldots$ is the neighbor sequence of peer y for the ID y, used during Join(y). Thus message m will be forwarded using the y-pointers at $y^j, \ldots, y^0 = y$ (if y^j has a y-pointer, then so does y^k for all $1 \leq k \leq j$) all the way down to peer y. If x^i does not have a y-pointer, it will simply forward the message m to x^{i+1}.

Given Invariant 1.1, peer x will locate peer y in the network if peer y is indeed part of the peer-to-peer system. The cost of routing to peer y can be bounded by:

Fact 1.7 *A message from peer x to peer y will be routed through a path with cost $O(\sum_{k=1}^{j}[c(x^{k-1}, x^k) + c(y^{k-1}, y^k)])$.*

The main challenge in the analysis of the PRR scheme is to show that this summation is indeed $O(c(x, y))$ in expectation.

A deficiency of this scheme, as described, is that there is a chance that we may fail to locate a pointer to y at x^1 through $x^{\log_b n}$. In this case, we must have more than one "root peer" in $T(y)$ (a root peer is a peer such that the length of its maximal prefix matching the ID of y is maximum among all peers in the network), and hence there is no guarantee that there exists a peer r which will have a global view of the network with respect to the ID of peer y. Fortunately, this deficiency may be easily rectified by a slight modification of the algorithm, as shown in [17].

The probability that the k-digit prefix of the label of an arbitrary peer matches a particular k-digit string $\gamma = \text{prefix}_k(y)$, for some peer y, is b^{-k}. Consider a ball B around peer x^{k-1} containing exactly b^k peers. Note that there is a constant probability (approximately $1/e$) that no peer in B matches γ. Thus the radius of B

is a lower bound (up to a constant factor) on the expected distance from x^{k-1} to its (k, y_k)-neighbor. Is this radius also an upper bound? Not for an arbitrary metric, since (for example) the diameter of the smallest ball around a peer z containing $b^k + 1$ peers can be arbitrarily larger than the diameter of the smallest ball around z containing b^k peers. However, it can be shown that the radius of B provides a tight bound on the expected distance from x^{k-1} to its (k, y_k)-neighbor. Furthermore, Equation 1.1 implies that the expected distance from x^{k-1} to its (k, y_k)-neighbor is geometrically increasing in k. The latter observation is crucial since it implies that the expected total distance from x^{k-1} to its (k, y_k)-neighbor, summed over all k such that $1 \leq k \leq j$, is dominated by (i.e., within a constant factor of) the expected communication cost from x^{j-1} to its (j, y_j)-neighbor x^j.

The key challenge of the complexity analysis of PRR is to show that, $c(x, y) = O(E[c(x^{j-1}, x^j)])$, and hence that the routing stretch factor in the PRR scheme is constant in expectation. This proof is technically rather involved and we refer the reader to [17]. In the next section, we will show how the PRR scheme can be elegantly modified in order to yield deterministic constant stretch. More specifically, the LAND scheme achieves deterministic stretch $1 + \epsilon$, for any fixed $\epsilon > 0$.

Updating the neighbor tables

In order to be able to have Join and Leave operations with low work complexity, while still enforcing low stretch Route operations and polylogarithmic degree, one needs to devise an efficient way for updating the neighbor tables upon the arrival or departure of a peer from the system. The PRR scheme alone does not provide such means. Luckily, we can combine the work by Hildrum et al. [12], which provides an efficient way for finding nearest neighbors in a dynamic and fully distributed environment, with the PRR scheme in order to be able to efficiently handle the insertion or removal of a peer from the system. Namely, teh work by Hildrum et al. presents an algorithm which can build the neighbor table of a peer p joining the system and update the other peers' neighbor tables to account for the new peer in the system with total work $O(\log^2 n)$.

4.2.2 The LAND scheme

The LAND scheme proposed by Abraham, Malkhi and Dobzinski in [2] is the first, and currently best-known, peer-to-peer overlay network design scheme to achieve constant deterministic stretch for routing, while maintaining a polylogarithmic diameter, degree, and work (for ROUTE, JOIN and LEAVE) for growth-bounded metrics. Note that LAND does not require a lower bound on the growth as PRR does. Like the PRR scheme, the LAND scheme was also designed for the more general problem of object location in peer-to-peer systems.

Namely, the main results of the LAND scheme are summarized in the following theorem:

Theorem 1.8 *The LAND scheme, when combined with a technique by Hildrum et al. for finding nearest neighbors, achieves an overlay peer-to-peer network with the following properties: deterministic $(1+\epsilon)$ stretch for* ROUTE *operations, for any fixed $\epsilon > 0$, $O(\log n)$ diameter, and expected $O(\log n)$ degree and work for* JOIN, LEAVE *and* ROUTE *operations.*

The LAND scheme is a variant of the PRR scheme. The implementation of ROUTE, JOIN, and LEAVE operations in this later scheme are basically the same as in PRR. The basic difference between the two schemes is on how the peer labels are assigned to the peers during the neighbor table construction phase. A peer may hold more than one label, some of which may not have been assigned in a fully random and independent way.

In a nutshell, the basic idea behind the LAND scheme is that instead of letting the distance between a peer x and its (i, α)-neighbor be arbitrarily large, it will enforce that this distance be always at most some constant β times b^i by letting peer x emulate a virtual peer with label γ such that $\text{prefix}_i(\gamma) = x_1 \ldots x_{i-1}\alpha$ if no peer has $x_1 \ldots x_{i-1}\alpha$ as a prefix of its label in a ball centered at x with $O(b^i)$ peers in it. Another difference in the LAND scheme which is crucial to guarantee a deterministic bound on stretch is that the set of "auxiliary" neighbors it maintains are only used during join/leave operations, rather than during routing operations such as in the PRR scheme.

The analysis of the LAND scheme is elegant, consisting of short and intuitive proofs. Thus, this is also a main contribution of this scheme, given that the analysis of the PRR scheme is rather lengthy and involved.

5 Other related work

There is a wealth of literature on peer-to-peer systems, and papers on this subject can be found in every major computer science conference. Peer-to-peer overlay networks can be classified into three categories: social networks, random networks, and structured networks.

Examples of social networks are Gnutella and KaZaA. Their basic idea of interconnecting peers is that connections follow the principle of highest benefit: a peer preferably connects to peers with similar interests by maintaining direct links to those peers that can successfully answer queries.

An example for random networks is JXTA, a Java library developed by SUN to facilitate the development of peer-to-peer systems. The basic idea behind the JXTA core is to maintain a random-looking network between the peers. In this way, peers are very likely to stay in a single connected component because random graphs are known to be robust against even massive failures or departures of nodes. Recent theory work on random peer-to-peer networks can be found in [15, 6].

Most of the scientific work on peer-to-peer networks has focused on structured overlay networks, i.e., networks with a regular structure that makes it easy to route in them with low overhead. The vast majority of these networks is based on the concept of virtual space. The most prominent among these are Chord [22], CAN [20], Pastry [7] and Tapestry [24]. The virtual space approach has the problem that it requires node labels to be evenly distributed in the space in order to obtain a scalable overlay network. Alternative approaches that yield scalable overlay networks for arbitrary distinct node labels are skip graphs [3], skip nets [8] and the hyperring [5].

As we have already seen above, structured overlay networks are also known that can take locality into account. All of this work is based on the results in [18, 17]. The first peer-to-peer systems were Tapestry [24] and Pastry [7]. Follow-up schemes addressed some of the shortcomings of the PRR scheme. In particular, the LAND scheme [2] improves the results in PRR as seen in the previous section; algorithms for efficiently handling peer arrival and departures are presented in [11, 12]; a simplified scheme with provable bounds for ring networks is given in [14]; a fault-tolerant extension is given in [10]; a scheme that addresses general networks, at the expense of an $O(\log n)$ stretch bound, is given in [19]; a scheme that considers object location under more realistic networks is given in [9]; and a first attempt at designing overlay networks in peer-to-peer systems consisting of mobile peers is presented in [1] (no formal bounds are proven in [1] for any relevant network distribution though).

Finally, overlay networks have not only been designed for wired networks but also for wireless networks. See, for example, [21] and the references therein for newest results in this area.

References

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *Proc. of DIALM-POMC*, 2004.

[2] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1+\epsilon)$ locality aware networks for DHTs. In *Proc. 9th ACM-SIAM Sympos. on Discrete Algorithms*, 2004.

[3] J. Aspnes and G. Shah. Skip graphs. In *Proc. of the 14th ACM Symp. on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[4] B. Awerbuch and C. Scheideler. Chord++: Low-congestion routing in chord. Unpublished manuscript, Johns Hopkins University, June 2003. See also http://www.in.tum.de/personen/scheideler/index.html.en.

[5] B. Awerbuch and C. Scheideler. The Hyperring: A low-congestion deterministic data structure for distributed environments. In *Proc. of the 15th ACM Symp. on Discrete Algorithms (SODA)*, 2004.

[6] C. Cooper, M. Dyer, and C. Greenhill. Sampling regular graphs and a peer-to-peer network. In *Proc. of the 16th ACM Symp. on Discrete Algorithms (SODA)*, 2005.

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

[8] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, 2003.

[9] Kirsten Hildrum, Robert Krauthgamer, and John Kubiatowicz. Object location in realistic networks. In *Proc. of ACM SPAA*, pages 25–35, 2004.

[10] Kirsten Hildrum and John Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. of DISC*, pages 321–336, 2003.

[11] Kirsten Hildrum, John Kubiatowicz, Sean Ma, and Satish Rao. A note on the nearest neighbor in growth-restricted metrics. In *Proc. of ACM Symp. on Discrete Algorithms (SODA)*, pages 560–561, 2004.

[12] Kirsten Hildrum, John Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. In *Proc. of ACM SPAA*, pages 41–52, 2002.

[13] David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proc. of ACM STOC*, pages 741–750, 2002.

[14] 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.

[15] P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *Proc. of the 17th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 155–164, 2005.

[16] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proc. of the 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 2003.

[17] C. Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32:241–280, 1999. A preliminary version of this paper appeared in *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, June 1997.

[18] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of ACM SPAA*, pages 311–320, 1997.

[19] R. Rajaraman, A. W. Richa, B. Vöcking, and G. Vuppuluri. A data tracking scheme for general networks. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures*, pages 247–254, July 2001.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM '01*, 2001.

[21] C. Scheideler. Overlay networks for wireless ad hoc networks. In *Proc. of the IMA Workshop on Wireless Communication, to appear*, 2005. See also http://www.in.tum.de/personen/scheideler/index.html.en.

[22] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. of the ACM SIGCOMM '01*, 2001.

[23] X. Wang, Y. Zhang, X. Li, and D. Loguinov. On zone-balancing of peer-to-peer networks: Analysis of random node join. In *Proc. of ACM SIGMETRICS*, 2004.

[24] B.Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, University of California at Berkeley, Computer Science Department, 2001.

Index