On sampling in higher-dimensional peer-to-peer systems

Goran Konjevod[⋆], Andréa W. Richa [⋆⋆], and Donglin Xia [⋆]

Department of Computer Science and Engineering,
Arizona State University, Tempe AZ 85287, USA,
{goran,aricha,dxia}@asu.edu

Abstract. We present fully distributed algorithms for random sampling of nodes in peer-to-peer systems, extending and generalizing the work of King and Saia [Proceedings of PODC 2004] from simple Chord-like distributed hash tables to systems based on higher-dimensional hierarchical constructions, like Content Addressable Networks (CAN). We also show preliminary results on the generalization of the problem to biased sampling. In addition, we provide an extension of CAN that requires only $O(1)$ space per node and achieves $O(\log n)$ lookup latency and message complexities.

1 Introduction

A distributed algorithm for random sampling of nodes in a peer-to-peer network provides a basic ingredient for the solution of several important problems, including load-balancing, Byzantine agreement and computing various statistics on data availability.

King and Saia [6] present a fully-distributed algorithm with expected logarithmic message complexity that with high probability (at least $1 - O(1/n)$) chooses a peer with *exactly* uniform distribution (i.e. with probability $1/n$). Both the expected latency (in terms of the number of links in the overlay network followed by the algorithm) and message complexity of their algorithm are logarithmic in the number of nodes. The algorithm does not assume any knowledge of the number of nodes n in the network. Their algorithm requires a peer-to-peer network with properties similar to those of Chord [11], in particular a one-dimensional circular keyspace.

In some applications, such as as peer-to-peer photo sharing and massively multiplayer games, multidimensional range-queries are critical, and the overlay network should support higher dimensional range queries [4]. Several structured overlay systems based on the geometry of two- or higher-dimensional space have been proposed, including CAN [9]. In this paper, we provide an efficient mechanism for *uniform random sampling in peer-to-peer overlay networks of higher*

[⋆] Supported in part by NSF grant CCR-0209138.
[⋆⋆] Supported in part by NSF CAREER grant CCR-9985284.

dimensions. Our work applies to peer-to-peer systems with properties similar to those of CAN.

The algorithm of King and Saia is based on a procedure to associate with each peer a part of the keyspace so that **(a)** the parts assigned to different peers are disjoint, **(b)** the measure of the part assigned to each peer is exactly the same, and **(c)** a constant fraction of the total keyspace is assigned to the peers. These properties reduce the problem of peer sampling to that of sampling of points in the keyspace, because it ensures that only an expected constant number of random keys must be sampled before one is generated that lies in a region belonging to some peer.

It may be possible to generalize this algorithm directly to multiple dimensions by assigning to peers disjoint regions of the keyspace. A natural idea would be to assign to each peer its region in the Voronoi diagram of the set of peer nodes. However, even on a line, the distances between consecutive points in a set of n uniformly generated points vary with high probability from $\Theta(1/n^2)$ to $\Theta(\ln n/n)$ [6]. Thus the Voronoi regions need to be patched to give each peer an equal-area region and the resulting structure very quickly becomes prohibitively complex, even in two dimensions.

The reader may already have noticed a direct reduction of our problem to the one-dimensional case: concatenate the coordinates of a node's ID (assumed to be binary sequences) to get a single binary sequence. If each coordinate is uniformly distributed, then so is the resulting sequence. These sequences define coordinates of the nodes on a circle (ring). Building an overlay network (say, Chord) on top of this circle allows one to directly use the algorithm of King and Saia. This effectively creates two overlays: a multi-dimensional one based on the original node coordinates, which allows multi-dimensional queries, and a one-dimensional one that allows random sampling.

There are two main problems with the approach above. First, it creates unnecessary overhead by the need of building and maintaining a second one-dimensional overlay network to be used just for sampling. Second, since the random sampling will be done via the auxiliary one-dimensional overlay network, which does not take into account the proximity of the nodes in the multidimensional space, the expected latency of the algorithm *may no longer be logarithmic* with respect to the *original* multidimensional overlay network.

Hence, in contrast to this simplistic approach, we use the Hilbert space-filling curve [10] to map the multi-dimensional keyspace into a circle, and a single multi-dimensional overlay (say, CAN) to implement a routing table and other functions useful for the peer-to-peer network, including those needed for the sampling algorithm itself. Our goal is to assign to each peer a part of the keyspace of equal volume, while keeping these parts disjoint and large enough to jointly cover a constant fraction of the whole keyspace. Thanks to the properties of the Hilbert curve, contiguous segments of the circle correspond to connected regions in the original keyspace. This means that any basic step of the algorithm that searches linearly through a bounded number of peers along the circle will only have to consider a bounded connected region in the original keyspace, thereby ensuring low latency (logarithmic in the number of nodes) in terms of the number of links followed in the multi-dimensional overlay network, unlike the simplistic solution above. In addition to this advantage, the use of a "native" overlay allows for the implementation of basic functions in the overlay network using the stronger geometric properties of the multidimensional space.

1.1 Our contributions

Given a fully decentralized peer-to-peer network in a multidimensional keyspace, we consider the problem of distributed random sampling of peer nodes. We assume that peers correspond to uniformly distributed points in a d-dimensional keyspace. We show that the one-dimensional sampling algorithm of King and Saia can be generalized to hierarchical systems with multidimensional keyspaces.

The sampling algorithm for the one-dimensional case relies on a basic function *next* that, given a peer, returns the next peer along the circle. In addition to this, the algorithm requires the ability to find the location of a given peer on the circle and to compute the measure of an interval between two given points. All we need assume is that the functions listed above can be computed efficiently by the peer-to-peer system, in other words, that the peer-to-peer system we use is compatible with the Hilbert curve. Our main result is stated in the theorem below:

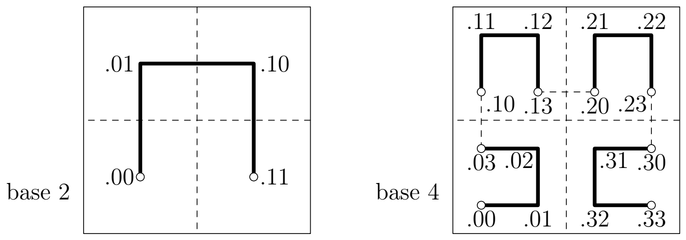
Theorem 1. *Given a peer-to-peer d-dimensional keyspace satisfying the properties above, the algorithm* **Choose A Random Peer** *selects each peer with probability exactly $1/n$, with high probability. The algorithm has expected latency $O(t_{lookup} + \log n)$ and sends $O(m_{lookup} + \log n)$ messages, where t_{lookup} and m_{lookup} are the latency and message complexities of* lookup. *In particular, a CAN can be implemented so that for any dimension d, $t_{lookup} = m_{lookup} = O(\log n)$, and $O(1)$ routing items in each peer are maintained for* lookup. *Therefore, in CAN the latency and message complexity of our algorithm are both $O(\log n)$ in expectation.*

2 Background and related work

2.1 The Hilbert curve

G. Peano discovered the first curve that passes through every point of a closed square [7], but one of the first graphical representations of such a space-filling curve was given by David Hilbert [5]. Space-filling curves are useful for reducing a multi-dimensional problem to a one-dimensional problem, and in this role have found uses in other contexts [3].

The Hilbert space-filling curve is a continuous mapping from the unit interval $[0, 1]$ onto the unit hypercube $[0, 1]^d$. It can be constructed recursively. First the d-dimensional cube is partitioned into 2^d congruent subcubes and accordingly, the unit interval into 2^d congruent subintervals. Then each subinterval is mapped onto a distinct subcube, and adjacent subintervals are mapped onto adjacent subcubes with a common facet. The example for 2 dimensions are presented in Figure 1(a). The same algorithm is applied to each subcube and its corresponding subinterval. The subinterval in each subcube is rotated and reflected so that it can connect to the preceding one to form a single continuous unit interval. Figure 1(b) shows the second step of the construction in 2 dimensions. Albert [2] provides a mathematical formalism for the Hilbert curve in arbitrary dimension d.



(a) First Step (b) Second Step

Fig. 1. The Generation of 2 Dimensional Hilbert Curve

We list several properties about the Hilbert curve in the following:

Proposition 1. *Each interval* $[0.b_1b_2..b_i, 0.b_1b_2..b_i + 2^{-i}]$ *of the curve fills a region of volume* 2^{-i}*, where* $i \in N$*, and* $b_j = 0, 1$*, for* $j = 1..i$*. We label this region by the* i*-digit binary sequence* $b_1b_2..b_i$*. Moreover, the region is a hyper-rectangle which can be split into two congruent sub-hyper-rectangles labeled by* $b_1b_2..b_i0$ *and* $b_1b_2..b_i1$*.*

In fact, we will think of the unit hypercube $[0, 1]^d$ as a wrapped d-torus, identifying for each coordinate i every pair of points $x, y \in [0, 1]^d$ such that $|x_i - y_i| = 1$ and $x_j = y_j$ for $i \neq j$, and the Hilbert curve $[0, 1]$ as a unit circle.

Proposition 2. *For any two regions labeled as* $A = a_1a_2..a_i$*,* $B = b_1b_2..b_j$*,* $(i \leq j)$*, if* A *is a prefix of* B*, then region* A *contains region* B*. If* A *is not a prefix of* B*, then the intersection of these two regions has volume of 0.*

Furthermore if $0.a_1a_2..a_i + 2^{-i} = 0.b_1b_2..b_j$*, or* $0.b_1b_2..b_j + 2^{-j} = 0.a_1a_2..a_i$*, i.e. their mapped intervals on the curve are connected, then regions* A *and* B *share a* $(d-1)$*-dimensional facet with positive* $(d-1)$*-dimensional volume.*

Proposition 3. *A random process of* n *points uniformly distributed on the Hilbert curve is equivalent to a random process of* n *points uniformly distributed in the unit hypercube according to the Hilbert curve mapping.*

2.2 CAN: a *d*-dimensional peer-to-peer network

CAN (Content-Addressable Network)[9] is a peer-to-peer system which takes a d-dimensional Cartesian coordinate space as the keyspace for its distributed hash table. The coordinate space is partitioned into hyper-rectangles, called zones, and each peer is responsible for a zone. Then a key is stored in the peer whose zone contains the key point. For example, Figure 2 shows a 2-dimensional $[0,1] \times [0,1]$ coordinate space with 6 peers.

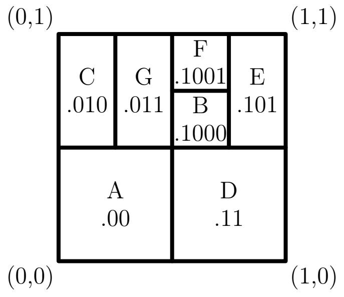


Fig. 2. A 2-Dimensional CAN with 6 Peers

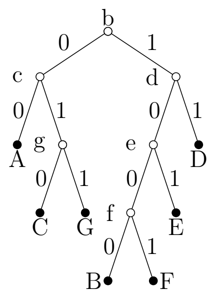


Fig. 3. The Binary Partition Tree for the CAN in Figure 2

Each peer maintains a routing table of all its neighbors in the coordinate space. Two peers are neighbors if they share a $(d-1)$-dimensional facet. Given a key, the lookup operation, or routing, in CAN is implemented by following the straight-line path through the coordinate space from the inquiring peer to the peer storing the key — that is, a peer will forward the lookup message through the peers responsible for the regions crossed by the respective straight-line path. Thus each peer maintains $2d$ neighbors and the average routing path length is $(d/4)(n^{1/d})$.

To join the network, a new peer first selects a random key point, and asks an existing peer to find the peer p who stores the key point. Then peer p will split its zone in half according to a given order of the dimensions, keeping one half of the zone assigned to itself and assigning the other half to the new peer.

We adapt the process by which CAN splits a zone to add a new peer to match the construction of the Hilbert curve. First, the whole key space $[0,1]^d$ can be partitioned into two zones labeled by 0 and 1. Then by Proposition 1, a zone assigned to a peer p and labeled by $b_1b_2..b_i$ can be partitioned into two equally-sized zones labeled by $b_1b_2..b_i0$ and $b_1b_2..b_i1$. The peer p will keep the half zone labeled by $b_1b_2..b_i0$ and assign the other half labeled $b_1b_2..b_i1$ to the new peer. Therefore we have the following lemma:

Lemma 1. *The whole key space is partitioned into zones labeled by binary sequences. The zone labeled by $b_1b_2..b_i$ is filled by an interval $[0.b_1b_2..b_i, 0.b_1b_2..b_i + 2^{-i}]$ of the Hilbert curve. Therefore these intervals also partition the Hilbert curve.*

Thus each zone is labeled by a unique binary sequence. We will also use the zone's label to identify the peer responsible for the zone. By Lemma 1, the

Hilbert curve is partitioned into intervals, each of which fills a distinct zone. Therefore, we have the following Lemma:

Lemma 2. *Given a peer labeled by $A = a_1a_2..a_i$, there exists one and only one peer labeled by $B = b_1b_2..b_j$ such that $0.a_1a_2..a_i + 2^{-i} = 0.b_1b_2..b_j$. Peers A and B are neighboring regions in the coordinate space.*

We call peer B the *next peer* of A, denoted by $B = next(A)$.

The routing scheme of CAN provided in [9] is a simple greedy routing in which the average routing path length is $(d/4)(n^{1/d})$. In fact, we can improve the expected routing path length to $O(\log n)$ by maintaining only $O(1)$ routing items at each node. We will maintain a binary tree that mimics the partitions performed by CAN[1].

We maintain a binary partition tree with n leaves according to the keyspace partition. Figure 3 illustrates the binary partition tree for the network shown in Figure 2. Each leaf of the binary tree corresponds to an existing zone (peer). Each inner node of the tree represents a zone that no longer exists, but was split at some previous time. The children of a tree node are the two zones into which it was split.

On the other hand, let each existing peer represent its corresponding leaf and the inner node that was split when the peer joined the network. Thus we have

Lemma 3. *The binary partition tree for CAN can be maintained with only $O(1)$ routing items in each peer.*

Now we show that the binary partition tree is well balanced.

Lemma 4. *The binary partition tree for CAN with n peers has $C_1 \log n \leq d_1(n) \leq d_2(n) \leq C_2 \log n$ with high probability, where $C_1 < 1$ and $C_2 > 2$ are constants, $d_1(n)$ is the distance from the root to the closest leaf, and $d_2(n)$ is the depth of the tree.*

Proof. (1) $C_1 \log n \leq d_1(n)$:Consider the full binary tree T with $C_1 \log n$ depth. The n^{C_1} leaves of tree T are considered as bins and the n peers as balls. Let X count the number of balls in a certain bin. Thus $E(X) = n^{1-C_1}$. By Chernoff bound, for any $0 < \delta < 1$, $Pr\{X < (1-\delta)E(X)\} < exp(-E(X)\delta^2/2)$. By setting $(1-\delta)E(X) = 1/2$, i.e. $\delta = 1 - 1/(2n^{1-C_1})$, we have

$$\begin{aligned} Pr\{X < 1/2\} &< exp(-n^{1-C_1}(1 - 1/(2n^{1-C_1}))^2/2) \\ &< exp(-n^{1-C_1}/8) \text{ (For } C_1 < 1,\ 1 - 1/(2n^{1-C_1}) > 1/2). \end{aligned} \tag{1}$$

Now the probability that there is a bin with less than 1/2 ball is less than $n^{C_1} \cdot e^{-n^{1-C_1}/8} < e^{C_1 \ln n - n^{1-C_1}/8}$, which can be arbitrarily small for any constant $C_1 < 1$, and n large enough. Thus with high probability the number of balls in

[1] Note that this binary tree need not replace the standard greedy routing algorithm of CAN (depending on the application, this may not be desirable). This binary tree will be used for finding a suitable routing path during random sampling operations.

every bin is larger than or equal to 1/2, thereby larger than or equal to 1. Thus $C_1 \log n \leq d_1(n)$.

(2) $d_2(n) \leq C_2 \log n$: Consider the full binary tree T with $C_2 \log n$ depth. The n^{C_2} leaves of tree T are considered as bins and the n peers as balls. Let X count the number of balls in a certain bin. Thus $E(X) = n^{1-C_2}$. By Chernoff bound, for any $\delta > 0$, we have $Pr\{X > (1+\delta)E(X)\} < (\frac{e}{1+\delta})^{(1+\delta)E(X)}$. By setting $(1+\delta) = 2n^{C_2-1}$, we have $Pr\{X \geq 2\} \leq (\frac{e}{2n^{C_2-1}})^2$. Thus the probability that there is a bin with more than or equal to 2 balls is less than or equal to $n^{C_2} \cdot (\frac{e}{2n^{C_2-1}})^2 = \frac{e^2}{4n^{C_2-2}}$, which can be arbitrarily small for any constant $C_2 > 2$ and n large enough. Therefore $d_2(n) \leq C_2 \log n$.

Therefore by maintaining such a binary partition tree we can achieve the average routing path length of $O(\log n)$ and $O(1)$ routing items in each peer.

3 Algorithm and Analysis

3.1 Estimating the number of peers

Since we want to choose each peer with the same probability, i.e., with probability $1/n$, where n is the number of peers in the network, it is clear that we must learn n in some sense. However, it is hard to count all the peers and keep n updated in a fully-distributed setting. Luckily, only an approximation to n is enough to sample peers uniformly. Before presenting the main algorithm, we first describe an algorithm by which a peer may estimate n to within a constant multiplicative factor. This is based on [6].

First the algorithm estimates n within a constant exponent by $\hat{n}_1$, the inverse of the volume of peer p's zone. Then it sums the volume of $c_1 \ln \hat{n}_1$ peers counting from p by the order of the *next* function. Finally n is estimated by the ratio of the number of these peers over their volume summation. The tightness of the estimation is determined by the constant c_1.

The algorithm is given as follows, where $vol(p)$ is the volume of peer p's zone, $next^{(s)}(p)$ means applying the *next* function s times starting from p, and $Vol(q,p)$ is the sum of the volumes of the peers from q to p, i.e.

$$Vol(q,p) = \sum_{\text{peer } r \text{ s.t. } 0.r \in [0.q, 0.p]} vol(r) \tag{2}$$

Estimate n

```
1.    $\hat{n}_1 \leftarrow vol(p)^{-1}$
2.    $s \leftarrow c_1 \ln \hat{n}_1$
3.    $t \leftarrow Vol(p, next^{(s)}(p))$
4.    Return $\hat{n}_2 \leftarrow s/t$.
```

We show that the above algorithm estimates n within a constant factor, basing our analysis on that of [6]. The main difference is that their result is based on the assumption that the peers are uniformly distributed in a unit circle, while in CAN each peer is responsible for a d-dimensional zone and cannot be abstracted as a point. Nevertheless, the peer in CAN is generated by a random key point, which allows us to generalize the analysis to work in our scenario. Note that by Proposition 3 a random point process in the curve is equivalent to a random point process in d-dimensions according to the Hilbert curve mapping. while this observation is helpful, we point out that this equivalence only makes our proof simpler, and that in CAN a random key point is just selected directly in d-dimensions.

When a peer arrives, a random key point is generated by the random process on the Hilbert curve. Therefore for a peer p in CAN, we define its **original point** as the point on the Hilbert curve which corresponds to its random key point.

We list the notations used throughout the paper:

– For a peer p, let $x(p)$ denote its original point on the Hilbert curve.
– For x, y on the unit circle (Hilbert curve), define the distance from x to y as $d(x, y) = y - x$ if $y \geq x$ and $d(x, y) = (1 + y) - x$ otherwise.
– For any interval I of the unit circle, denote by $num_x(I)$ the number of random points in I.
– For any interval I in the unit circle, denote by $num_p(I)$ the number of peers s such that $0.s \in I$.
– For given functions $f(n)$ and $g(n)$, we say that $f(n)$ is a (γ_1, γ_2) *approximation* of $g(n)$ if $\gamma_1 g(n) \leq f(n) \leq \gamma_2 g(n)$.
– For any two peers p and q, the number of peers from q to p is given by $num(q, p) = |\{\text{peer } r \text{ s.t. } 0.r \in [0.q, 0.p]\}|$

Since the original points of peers in CAN are considered as a random point process in the unit circle, we can relate the peers in CAN to their original points and generalize the original results.

Lemma 5. *For any two neighboring peers p and q, (e.g. $q = next(p)$), let X be the number of original points in $[0.p, 0.q]$. Then $E(X) = 1 - vol(p)$.*

Proof. Let $vol(p) = 2^{-t}$. Let Z_i be the zone with volume 2^{-i} that contains the zone p, for $i = 0, \ldots, t-1$. Note that zone Z_i is split into half when a new peer joins and the random point falls in the region of zone Z_i. Let x_i be a variable to indicate whether a random point falls in the region of zone p when zone Z_i is split. Then $E(x_i) = 2^{-t}/2^{-i} = 2^{i-t}$. Thus $E(X) = \sum_{i=0}^{t-1} E(x_i) = 1 - 2^{-t} = 1 - vol(p)$.

For reference, we list the lemma from [6] that we use in our proof:

Lemma 6. *[6] If n points are distributed uniformly at random in the unit circle, let $\alpha_1, \alpha_2, \epsilon$ be fixed positive constants with $\alpha_1 < \alpha_2$ and $0 \leq \epsilon \leq 1/2$. Let $C > 144/(\alpha_1 \epsilon^2)$. Then for two any interval I on the unit circle such that $C\alpha_1 \ln n \leq num_x(I) \leq C\alpha_2 \ln n$, we have $C(1-\epsilon)\alpha_1(\ln n/n) \leq |I| \leq C(1+\epsilon)\alpha_2 \ln n/n$ with probability at least $1 - 1/n$.*

Lemma 7. *Let $\alpha_1, \alpha_2, \epsilon$ be fixed positive constants with $\alpha_1 < \alpha_2$ and $0 \leq \epsilon \leq 1/2$. Let $C > 144/(\alpha_1 \epsilon^2)$. Then for any two peers p and q such that $C\alpha_1 \ln n \leq num(p, q) \leq C\alpha_2 \ln n$, we have $\frac{1}{2} \cdot C(1-\epsilon)\alpha_1(\ln n/n) \leq Vol(p, q) \leq 3 \cdot C(1 + \epsilon)\alpha_2 \ln n/n$ with probability at least $1 - 1/n$.*

Proof. Let $I = [0.p, 0.p + Vol(p,q)]$. Since the next peer t of q has $0.t = 0.q + vol(q) = 0.p + Vol(p,q)$, we have $I = [0.p, 0.t]$. Let $num_p(I)$ be the number of peer s such that $0.s \in I$, i.e. $num_p(I) = num(p,t)$. Then $num_p(I) = num(p,q) + 1$. Let peer r be the first peer laid down in the network such that $0.r \in [0.p, 0.t]$.

(1) $\frac{1}{2} \cdot C(1-\epsilon)\alpha_1(\ln n/n) \le Vol(p,q)$

Let $I' = [2 \cdot 0.p - 0.r, 2 \cdot 0.t - 0.r]$. Then $|I'| = 2|I|$. For any peer s such that $0.s \in [0.p, 0.t]$, $s \ne r$, we have $|x(s) - 0.r| \le 2|0.s - 0.r|$. Then $x(s) \in I'$ since $0.s \in [0.p, 0.t]$. Thus $num_x(I') \ge num(p,q) + 1 - 1 \ge C\alpha_1 \ln n$. Thus by Lemma 6, $|I'| \ge C(1-\epsilon)\alpha_1(\ln n/n)$. Therefore $Vol(p,q) = |I| \ge \frac{1}{2} \cdot C(1-\epsilon)\alpha_1(\ln n/n)$.

(2) $Vol(p,q) \le 3 \cdot C(1+\epsilon)\alpha_2 \ln n/n$

Let $I' = [0.p', 0.t']$ where p' and t' be the first peer laid down in the network such that $0.p' \in [0.p, 0.r)$ and $0.t' \in (0.r, 0.t]$ respectively. Then $|I'| \ge \frac{|I|}{2}$.

Let $num_x(I') = x_1 + x_2$, where x_1 is the number of such peers s that $x(s) \in I'$ and $0.s \in I'$, and x_2 is the number of such peers s that $x(s) \in I'$ but $0.s \notin I'$. First we have $x_1 \le num_p(I') \le num_p(I) \le C\alpha_2 \ln n$. Then since all the peers s such that $0.s \in (0.p', 0.r)$ or $0.s \in (0.r, 0.t')$ should have $x(s) \in [0.p', 0.r]$ or $x(s) \in [0.r, 0.t']$ respectively, by Lemma 5 we have $E(x_2) < (1 - (0.r - 0.p')) + (1 - (0.t' - 0.r)) = 2 - |I'| < 2$. Thus by Chernoff bound, $Pr\{x_2 \ge \frac{1}{2} \cdot C\alpha_2 \ln n\} \le 2^{-\frac{1}{2} \cdot C\alpha_2 \ln n} = n^{-\frac{1}{2} \cdot C\alpha_2 \ln 2}$. Thus with probability $1 - O(1/n)$, we have $num_x(I') = x_1 + x_2 \le \frac{3}{2} \cdot C\alpha_2 \ln n$. Thus by Lemma 6, $|I'| \le \frac{3}{2} \cdot C(1+\epsilon)\alpha_2 \ln n/n$. Therefore $Vol(p,q) = |I| \le 3 \cdot C(1-\epsilon)\alpha_1(\ln n/n)$.

Lemma 8. *With probability at least* $1 - 2/n$*, the algorithm 'Estimate n' ensures that* $(1/6 - \epsilon_1)n \le \hat{n}_2 \le 6 + \epsilon_1$*, for any positive constant* ϵ_1 *and* n *sufficiently large.*

Proof. Since for a peer p, $\log(vol(p)^{-1})$ is the depth of p's leaf in the binary partition tree, by Lemma 4, $\log(vol(p)^{-1})$ is an (α, β) approximation to $\log n$ for any fixed constants $\alpha < 1$ and $\beta > 2$. Thus $s = c_1 \ln vol(p)^{-1}$ is an (α, β) approximation to $c_1 \ln n$. Similarly, Lemma 7 shows that t in our algorithm is a $(\alpha/2 - \epsilon, 3\beta + \epsilon)$ approximation to $(c_1 \ln n)/n$ for any $\epsilon > 0$, for n and c_1 sufficiently large. Thus, $\hat{n}$ is a $(\frac{\alpha}{3\beta+\epsilon}, \frac{\beta}{\alpha/2-\epsilon})$ approximation to n for c_1 and n sufficiently large.

3.2 Choosing a Random Peer

By Lemma 8, we can estimate the number of peers as $\hat{n}_2$, a (γ_1, γ_2)-approximation to n, for constants γ_1, γ_2. Then let $n' = \hat{n}_2/\gamma_1$, and $\lambda = 1/(13n')$. Thus $n' \ge n$, $\lambda \le 1/(13n)$ and $\lambda = \Theta(1/n)$.

Our algorithm for choosing a random peer works as follows. First it randomly selects a key x in the key space $[0,1]^d$, looks up the peer p who stores the key and picks up a random number T in $[0, vol(p)]$. Then if there is a peer p such that $Vol(lookup(x), p) - (vol(lookup(x)) - T) \le \lambda \cdot num(lookup(x), p)$ with $num(lookup(x), p) \le 12 \ln n'$, the algorithm returns the first such peer. Else it repeats until a peer is returned. We will show that the expected number of repetitions of the *while* loop is $O(1)$ with high probability. The algorithm is a direct generalization of the one given by King and Saia [6].

Choose Random Peer

```
1.   While TRUE do:
2.      $x \leftarrow$ random number in $[0,1]^d$;
3.      $p = lookup(x)$; $T \leftarrow$ random number in $[0, vol(p)]$;
4.      For $(i = 1; i \leq 12 \ln n'; i++)$
5.         If $(T \leq i \cdot \lambda)$ return $p$;
6.         Else
7.            $p = next(p)$;
8.            $T = T + vol(p)$;
```

Definition 1. *For any peer labeled as p, let* ***first(p)*** *be the first peer such that the sum of the volumes of the peers from $first(p)$ to p including $first(p)$ and p is larger than or equal to the number of these peers multiplied by λ, i.e. $Vol(first(p), p) \geq \lambda num(first(p), p)$.*

Lemma 9. *For any peer p, if $num(first(p), p) \leq 12 \ln(n')$, the algorithm will choose p with probability λ in each iteration of the* ***while*** *loop.*

Proof. Let $q = first(p)$. Note that if peer $s = lookup(x)$ is not in $[0.q, 0.p]$, then p couldn't be returned by the algorithm. Otherwise assume $0.s \notin [0.q, 0.p]$ and p is returned. Denote t the previous peer of $first(p)$, i.e. $next(t) = first(p)$. Since t is visited by algorithm before p, then $T_1 > \lambda num(s,t)$, where T_1 is the value of T at the time t is visited. Then when p is visited, $T = T_1 + Vol(first(p), p) > \lambda num(s,t) + \lambda num(first(p), p) = \lambda num(s, p)$, which contradicts the condition to return p.

Then we argue by induction on $k = num(first(p), p)$.

Base: $k = 1$, i.e. $vol(p) \geq \lambda$. The probability that p is selected is $\frac{vol(p)}{1} \cdot \frac{\lambda}{vol(p)} = \lambda$.

Induction Step: For $k > 1$, assume peer p will be chosen if $num(first(p), p) < k$. Now consider the case $num(first(p), p) = k$.

Denote $q = first(p)$. Then for any peer s such that $0.s \in [0.q, 0.p)$, $first(s)$ should also be in $[0.q, 0.p)$. Otherwise by the definition of $first()$, $Vol(q, s) < \lambda num(q, s), Vol(next(s), p) < \lambda num(next(s), p)$. Thus $Vol(q, p) < \lambda num(q, p)$, which contradicts $q = first(p)$. Therefore all such s have $first(s)$ in $[0.q, 0.p)$ and by the induction hypothesis they will be selected with probability λ. According to our algorithm, if the peer $s = lookup(x)$ is in $(0.q, 0.p]$, then one of the peers in $(0.q, 0.p]$ must be selected since p is a candidate due to $first(p) = q$. And if the peer $s = lookup(x)$ is equal to q, then one of the peers in $[0.q, 0.p]$ will be selected with probability $(\lambda num(q, p) - Vol(next(q), p))/vol(q)$.

Therefore the probability that one of the peers in $[0.q, 0.p]$ will be selected is given as

$$\frac{Vol(q,p)}{1} \cdot \frac{Vol(next(q),p) + \frac{\lambda num(q,p) - Vol(next(q),p)}{vol(q)} \cdot vol(q)}{Vol(q,p)} = \lambda num(q,p) \quad (3)$$

Since every peer $s \in [0.q, 0.p]$ other than p has probability λ of being selected by the induction hypothesis, peer p must be selected with probability λ.

Lemma 10. *With probability greater than* $1 - 1/n$*, for any peer* p *and* q *such that* $num(p, q) > 12 \ln n$*, we have* $Vol(p, q) > \ln n/n$*.*

Proof. Let $I = [0.p, 0.p + Vol(p, q)]$. Denote $t = next(q)$. Then $I = [0.p, 0.t]$ and $num(p, t) = num(p, q) + 1$. Let peer r be the first peer laid down in the network such that $0.r \in [0.p, 0.t]$. Let $I' = [2 \cdot 0.p - 0.r, 2 \cdot 0.t - 0.r]$. Then $|I'| = 2|I|$. Note that for any peer s such that $0.s \in [0.p, 0.t]$, $s \neq r$, its original point will fall in the interval I'. Thus $num_x(I') \geq num(p, q) + 1 - 1 > 12 \ln n$. By a simply application of Chernoff bound, we have any interval containing more than $6 \ln n$ points has length greater than $\ln n/n$[6]. Thus $|I'| > 2 \ln n/n$. Therefore $Vol(p, q) = |I| > \ln n/n$.

Lemma 11. *With probability greater than* $1 - 1/n$*, for any peer* p*,*
$num(first(p), p) \leq 12 \ln n'$*.*

Proof. Let $q = first(p)$. By contradiction, assume $num(first(p), p) > 12 \ln n'$. Let s be the peer such that $0.s \in (0.q, 0.p]$ and $num(s, p) = 12 \ln n'$. Then by Lemma 10, $Vol(s, p) \geq \ln n/n \geq \ln n'/n' \geq 12 \ln n'/(13n') = \lambda \cdot num(s, p)$. Then $first(p)$ must be in $[0.s, 0.p]$, which contradicts $q = first(p)$ and $0.s \in (0.q, 0.p]$. Therefore $num(first(p), p) \leq 12 \ln n'$.

Theorem 2. *With probability at least* $1 - 3/n$*, each peer is chosen with probability exactly* $1/n$*.*

Proof. By Lemma 11, for any peer p, $num(first(p), p) \leq 12 \ln n'$. Then by Lemma 9, our algorithm will choose p with probability λ in each iteration of the *while* loop. Since $\lambda = \Theta(1/n)$, the expected number of the repetitions of the *while* loop will be $\Theta(1)$. Therefore each peer will be chosen with the same probability.

3.3 Latency and Message Complexity

Proof of Theorem 1: By Theorem 2, our algorithm selects each peer with probability $1/n$, with high probability. Now consider its latency and message complexity.

Since our algorithm for estimating n takes $O(\log n)$ *next* operations in expectation, it has expected latency $O(\log n)$ and message complexity $O(\log n)$.

For each iteration of the *while* loop of the algorithm, there is one *lookup* operation and at most $O(\log n)$ *next* operations. By Lemma 4 and 3, the lookup operations in CAN implemented with the aid of the partition tree have expected complexity $O(\log n)$, with $O(1)$ routing items being maintained at each peer.

Therefore, in CAN, our algorithm has expected latency $O(\log n)$ and sends $O(\log n)$ messages in expectation. ■

4 Future work: biased distributions and more general networks

In this section, we propose two natural generalizations of the uniform random sampling problem which pose interesting lines of future work.

First we consider the problem of handling more general sampling distributions and to choose peers with a biased probability. In other words, we would like to choose a peer p with probability proportional to $f(p)$, thereby with probability $f(p)/\sum f(x)$, where $f(p)$ can be any positive function on peer p. A case that is easy to solve is when $\max(f(x))/\min(f(x)) = \Theta(1)$. We can estimate $\sum f(x)$ as σ to a constant factor with a technique similar to that we used for estimating n. We assign each peer a region of volume $\lambda(p) = c \cdot f(p)/\sigma$, where c is a constant. Since $\max(f(x))/\min(f(x)) = \Theta(1)$ and σ approximates $\sum f(x)$ within a constant factor, we have $\lambda(p) = \Theta(1/n)$ for all peers p. Then the sampling algorithm still works if Line 5 is replaced by '**If** $(T \leq \Lambda_i)$ **return** p;', where $\Lambda_i = \sum_{k=0}^{i-1} \lambda(next^{(k)}(p))$. For brevity, we omit the proof of correctness. Dealing with distributions that are not "almost uniform" as described above seems to be rather more difficult.

The second generalization of the random sampling problem that we consider is to devise efficient algorithms for selecting a peer uniformly at random in other overlay peer-to-peer systems, such as the locality-aware systems of Plaxton et al. [8] and Abraham et al. [1], or in systems that provide even less structure than Chord or CAN. Our approach of using a space-filling curve can be used in most systems based on peers embedded in geometric space.

References

1. I. Abraham, D. Malkhi, O. Dobzinski: *LAND: Stretch* $(1 + \epsilon)$ *Locality Aware Networks for DHTs.* In ACM-SIAM SODA, 2004.
2. J. Alber, R. Niedermeier: *On Multi-Dimensional Hilbert Indexings.* In 4th COCOON, 1998.
3. L. K. Platzmann, J. J. Bartholdi: *Spacefilling curves and the planar travelling salesman problem.* In Journal of the ACM, 36(4):719-737, Oct 1989.
4. P. Ganesan, B. Yang, H. Garcia-Molina: *One torus to rule them all: multi-dimensional queries in P2P systems.* In 7th WebDB, 2004.
5. D. Hilbert: *Über stetige Abbildung einer Linie auf ein Flachenstuck.* Math. Annln., 38:459-460, 1891.
6. V. King, J. Saia: *Choosing a random peer.* In Proc. of the ACM PODC, 2004.
7. G. Peano: *Sur une courbe qui remplit toute une aire plane.* Math. Annln., 36:157-160, 1890.
8. C. Plaxton, R. Rajaraman, A. Richa: *Accessing nearby copies of replicated objects in a distributed environment.* In Proc. of the ACM SPAA, 1997.
9. S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker: *A scalable content-addressable network.* In Proc. of the ACM SIGCOMM, 2001.
10. H. Sagan: *Space-filling curves*, Springer Verlag 1994.
11. I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan: *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.* In ACM SIGCOMM, 2001.