A Probabilistic Analysis of Kademlia Networks*

Xing Shi Cai and Luc Devroye

School of Computer Science, McGill University of Montreal, Canada,
xingshi.cai@mail.mcgill.ca,
lucdevroye@gmail.com

Abstract. Kademlia [3] is currently the most widely used searching algorithm in P2P (peer-to-peer) networks. This work studies an essential question about Kademlia from a mathematical perspective: how long does it take to locate a node in the network? To answer it, we introduce a random graph $\mathcal{K}$ and study how many steps are needed to locate a given vertex in $\mathcal{K}$ using Kademlia's algorithm, which we call the *routing time*. Two slightly different versions of $\mathcal{K}$ are studied. In the first one, vertices of $\mathcal{K}$ are labeled with fixed IDs. In the second one, vertices are assumed to have randomly selected IDs. In both cases, we show that the routing time is about $c \log n$, where n is the number of nodes in the network and c is an explicitly described constant.

1 Introduction to Kademlia

A P2P (peer-to-peer) network [4] is a computer network which allows sharing of resources like storage, bandwidth and computing power. Unlike traditional client-server architectures, in P2P networks, a participating computer (a *node*) is not only a consumer but also a supplier of resources. Nowadays, major P2P services in the internet often have millions of users. For an overview of P2P networks, see Steinmetz [5].

The huge size of P2P networks raises one challenge—among millions of nodes, how can a node find another one efficiently? To address this, a group of algorithms called DHT (Distributed Hash Table) [6] was invented in the early 2000s, including Pastry [7], CAN [8], Chord [9], Tapestry [10], and Kademlia [3]. Created by Maymounkov and Mazières in 2002, Kademlia has become the de facto standard searching algorithm for P2P services. It is used by BitTorrent [11] and the Kad network [12], both of which have more than a million nodes.

Kademlia assigns each node an ID chosen uniformly at random from $\{0,1\}^d$, the d-dimension hypercube, where d is usually 128 [12] or 160 [11]. Thus we *always* refer to a node by its ID. Given two IDs $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$, Kademlia defines their XOR *distance* by

$$\delta(x, y) = \sum_{i=1}^{d} (x_i \oplus y_i) \times 2^{d-i},$$

* We would like to thank Carlton Davis, José M. Fernandez, (authors of [1] and [2]) and Mahshid Yassaei for collaborations and explanations regarding Kademlia.

where $\oplus$ denotes the XOR operation

$$u \oplus v = \begin{cases} 1 & \text{if } u \neq v, \\ 0 & \text{otherwise.} \end{cases}$$

Note that distance and closeness *always* mean XOR distances between IDs in this work.

Since it is almost impossible for a node to know where all other nodes are located, in Kademlia a node, say x, is only responsible for maintaining a table (a *routing table*) for a small number of other nodes (x's *neighbors*). Roughly speaking, $\{0,1\}^d$ is partitioned into d subsets, such that nodes in the same subset have similar distances to x. Within each subset, up to k nodes' information is recorded in a list (a *k-bucket*), where k is a constant which usually equals 8 [11], 10 [12] or 20 [13]. All of x's k-buckets form x's routing table.

When x needs to locate node y which is not in its routing table, x sends queries to α of its neighbors which are closest to y, where α is a constant, sometimes chosen as 3 [14] or 10 [13]. A recipient of x's message returns locations of k of its own neighbors with shortest distance to y. With this information, x again contacts α nodes that are closest to y. This approach (*routing*) repeats until no one closer to y can be found. The efficiency of routing is critical for the overall performance of Kademlia. Its analysis is the topic of this work.

2 Our Model

Consider a Kademlia network of n nodes $X_1, \ldots, X_n$. Writing IDs as strings consisting of zeros and ones, from higher bits to lower bits, we can completely represent $X_1, \ldots, X_n$ in a binary *trie*, as depicted in Fig. 1. A trie is an ordered tree data structure invented by Fredkin [15]. For more on tries, see Szpankowski [16]. Paths are associated with bit strings—0 corresponds to a left child, and 1 to a right child. The bits encountered on a path of length d to a leaf is the ID, or value, of the leaf. In this manner, the binary trie, has height d, and precisely n leaves of distance d from the root.

Let $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ be two IDs (leaves) in the trie. Let $\ell(x, y)$ be the length of the path from the root to x and y's lowest common ancestor, i.e., the length of x and y's common prefix. We have

$$\ell(x, y) = \max\{i : x_1 = y_1, \ldots, x_i = y_i\} \ .$$

It is easy to verify that $\ell(x, y)$ bounds the distance of x and y by

$$2^{d-\ell(x,y)-1} \leq \delta(x, y) < 2^{d-\ell(x,y)} \ .$$

Thus if we partition $\{0,1\}^d \setminus \{x\}$ according to distances to x as follows

$$\mathcal{D}_i(x) = \{y : 2^{i-1} \leq \delta(x, y) < 2^i\}, \quad i = 1, \ldots, d,$$

then $\mathcal{D}_i(x)$ is equivalent to a subtree in the trie, in which each node shares a common prefix of length i with x. Therefore, we have the equivalent definition

$$\mathcal{D}_i(x) = \{y : \ell(x, y) = d - i\}, \quad i = 1, \ldots, d \ .$$

Let $\mathcal{B}_i(x)$ be the set of IDs in the k-bucket of x corresponding to $\mathcal{D}_i(x)$. In our model, we assume that if $|\mathcal{D}_i(x)| \leq k$, then $\mathcal{B}_i(x) = \mathcal{D}_i(x)$. Otherwise, for all $A \subset \mathcal{D}_i(x)$ with $|A| = k$, we have

$$\mathbb{P}\left\{\mathcal{B}_i(x) = A\right\} = \binom{|\mathcal{D}_i(x)|}{k}^{-1} \ .$$

In other words, we fill up each k-bucket uniformly at random without replacement.

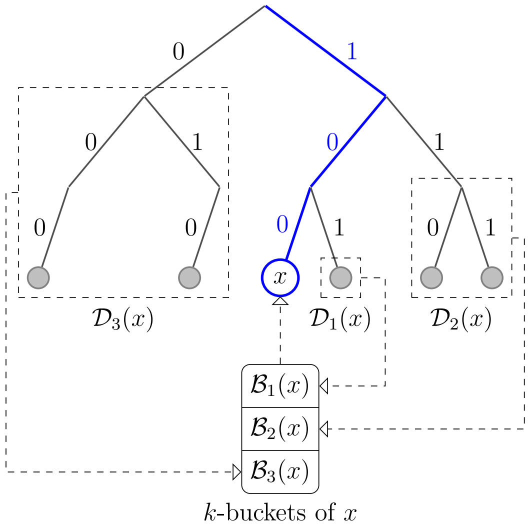

Fig. 1. An example of Kademlia ID trie and k-buckets. Given an ID $x = (1, 0, 0)$, the trie is partitioned into subtrees $\mathcal{D}_1(x), \mathcal{D}_2(x), \mathcal{D}_3(x)$. Node x maintains a k-bucket for each of these subtrees containing the information of up to k nodes in the subtree, which we denote by $\mathcal{B}_1(x), \mathcal{B}_2(x), \mathcal{B}_3(x)$ respectively.

Consider a directed graph $\mathcal{K}$ with vertex set $\mathcal{V} = \{X_1, \ldots, X_n\}$. Let its edge set be

$$\mathcal{E} = \{(u, v) : u, v \in \mathcal{V}, \ v \in \cup_{i=1}^{d} \mathcal{B}_i(u)\} \ .$$

Put differently, we add a directed edge (u, v) in $\mathcal{K}$ if and only if v is in one of u's k-buckets. When $\alpha = 1$, only one node is queried at each step of routing, the search process starting at x for y can be seen as a path ρ_{xy} in $\mathcal{K}$ (the *routing path*). It starts from vertex x, then jumps to the vertex that is closest to y among all x's neighbors. From there, it again jumps to the adjacent vertex that is closest to y. Let y^* be the unique vertex with the shortest distance to y in $\mathcal{K}$. Since the distance between the current vertex and y^* decreases at each step until zero, ρ_{xy} has no loop and always ends at y^*. In fact, $\mathcal{K}$ is always strongly connected.

Let T_{xy} be the path length of ρ_{xy}. Since T_{xy} equals the number of rounds of messages x needs to send before routing ends, we call it the *routing time.* Our first main result assumes that $X_1 = x_1, \ldots, X_n = x_n$, where $x_1, \ldots, x_n$ are all fixed d-bit vectors, which we call the *deterministic* ID *model.* The randomness thus comes only from the filling of the k-buckets. We always assume that $d \geq \log_2 n$, and we let $n \to \infty$ (and thus $d \to \infty$) in our asymptotic results. (Note that in this work $\log n$ denotes the natural logarithm of n.)

Theorem 1. *In the deterministic* ID *model, we have*

$$\sup_{x_1,\ldots,x_n} \sup_x \sup_y \mathbb{E}\left[T_{xy}\right] \leq (c_k + o(1)) \log n,$$

$$\sup_{x_1,\ldots,x_n} \sup_x \mathbb{E}\left[\sup_y T_{xy}\right] \leq (c'_k + o(1)) \log n,$$

$$\sup_{x_1,\ldots,x_n} \mathbb{E}\left[\sup_x \sup_y T_{xy}\right] \leq (c^*_k + o(1)) \log n,$$

where c_k, c'_k, c^*_k are constants depending only on k. In particular, we have $c_k = 1/H_k$, where $H_k = \sum_{i=1}^{k} 1/i$, also known as the k-th harmonic number.

The first inequality in this theorem gives an upper bound over the expected routing time between two fixed nodes. Since $c_k \leq c_1 \leq 1/\log 2$, the first bound is better than the $\lceil \log_2 n \rceil$ bound described by Maymounkov and Mazières in the original Kademlia paper [3]. The second inequality considers the expectation of the maximal routing time when the starting node is fixed, and a look-up is performed for each of the n destinations. The third one considers the expectation of the maximal routing time in the whole network if all n nodes were to look up all n destinations.

Our second main result considers the situation when $X_1, \ldots, X_n$ are selected uniformly at random from $\{0,1\}^d$ without replacement (the *random* ID *model*). Given an ID x, let $\widehat{x}$ denote the ID that is farthest away from x (x's *polar opposite*). Since by symmetry $T_{X_1\widehat{X}_1}, T_{X_2\widehat{X}_2}, \ldots T_{X_n\widehat{X}_n}$ are identically distributed, we only need to study $T_{X_1\widehat{X}_1}$, which we denote by $\widehat{T}$.

Theorem 2 ([17]). *In the random* ID *model, we have*

$$\frac{\widehat{T}}{\log n} \to \frac{1}{g(k)} \qquad \text{in probability,}$$

as $n \to \infty$, where $g(k) = H_k + O(1)$ is a function of k.

Since $\widehat{T} = T_{X_1\widehat{X}_1}$, we see that

$$\frac{1}{g(k)} \leq \frac{1}{H_k} \ .$$

However, we have almost identical behavior because $g(k) = H_k + O(1)$ as $k \to \infty$. Let $d = d(n) \geq \log_2 n$. By the probabilistic method, Theorem 2 implies that for every $\epsilon > 0$, there exists for every n a non-repetitive sequence $(x_1(n), \ldots, x_n(n)) \in (\{0,1\}^d)^n$ of deterministic IDs such that with probability going to one,

$$T_{X_1 \widehat{X}_1} \geq \left(\frac{1}{g(k)} - \epsilon \right) \log n \ .$$

In view of $g(k) = H_k + O(1)$, we thus see that the first bound of Theorem 1 is almost tight.

Recall that y^* denotes the node that is closest to ID y. If we look at the lowest common ancestor of y^* and each hop of ρ_{xy}, then searching y from x can be seen as travelling downwards along the path from the root to the leaf y^* in the trie, with the distance of each hop being random. In the random ID model, when n is large, we can approximate this process in a full binary trie with infinite depth. In Sect. 4, we sketch the proof of Theorem 2 which uses this method. This does not work in the deterministic ID model as the structure of the ID trie can be unbalanced. But in Sect. 3, we show that there is another way to bound the routing time.

Due to its success, Kademlia has aroused great interest among researchers. But this is the first time that it is studied from a mathematical perspective. Our results point out one important reason for the success of Kademlia — its routing algorithm, while being extremely simple, works surprisingly well. This work also shows that probabilistic methods together with the right choice of a data structure, a trie in our case, could significantly simplify the analysis of a problem which was previously considered too troublesome to analyze rigorously.

3 The Deterministic ID Model

In this section, we assume that $X_1 = x_1, \ldots, X_n = x_n$, where $x_1, \ldots, x_n$ are fixed d-bits vectors. Note that the distribution of T_{xy} is decided only by distances between vertices and the distance between x and y. Thus, by rotating the hypercube, we can always assume y to be a specific ID, which we choose $\bar{1} = (1, 1, \ldots, 1)$.

Figure 2 depicts the first hop of $\rho_{x\bar{1}}$ as jumping from one leaf to another in the ID trie. It is easy to see that if we always arrange branches representing 1 to the right hand side, which we take as a convention, then the closer a leaf is to the right, the closer it is to $\bar{1}$. Thus the rightmost leaf in the trie, which we always denote by y', is closest to $\bar{1}$ and is thus the end point of $\rho_{x\bar{1}}$.

Write $\rho_{x\bar{1}} = (z_0, z_1, \ldots)$ where $z_0 = x$. Let $i = d - \ell(z_0, \bar{1})$. We can see from Fig. 2 that z_1, the first hop, must belong to $\mathcal{D}_i(z_0)$, the highest subtree on the right hand side of z_0, which we denote by S_0. Since being z_0's neighbor implies membership in one of z_0's k-buckets, we have $z_1 \in \mathcal{B}_i(z_0) \subseteq S_0$. Recalling how $\mathcal{B}_i(z_0)$ is decided, we can think of the first hop as selecting up to k leaves from S_0 uniformly at random and choosing the rightmost one as z_1. Thus we can define of $\rho_{x\bar{1}}$ recursively as follows:

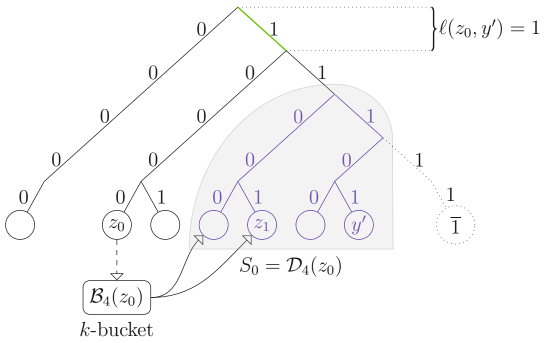

Fig. 2. An example of the first hop of $\rho_{x\bar{1}}$ with $d = 5$, $k = 2$. Since $d - \ell(z_0, y') = 4$, z_1 must be in subtree $S_0 = \mathcal{D}_4(z_0)$, which is the highest subtree to the right of z_0. We choose up to k leaves from S_0 uniformly at random without replacement, and let z_1 be the rightmost one.

– Let $z_0 = x$. Repeat the following step.
– Given z_t and $t \geq 0$, let S_t be highest subtree on the right hand side of z_t. If $S_t = \emptyset$, terminate. Otherwise select up to k leaves from S_t uniformly at random without replacement, and let the rightmost one be z_{t+1}.

Since $S_0 \supset S_1 \supset \cdots \supset S_{T_{x\bar{1}}} = \emptyset$, by studying how quickly the sequence $(|S_t|)_{t\geq 0}$ decreases to 0, we can bound how big $T_{x\bar{1}}$ could be. Although it is difficult to write the distribution of $(|S_t|)_{t\geq 0}$, we can approximate it with another sequence $(W_t)_{t\geq 0}$. Let $B(k)$ be the minimum of k independent uniform $[0,1]$ random variables. Let $(B_t)_{t\geq 0}$ be a sequence of i.i.d. random variables with distribution $B(k)$. We define $W_t = |S_0| \times \prod_{s=1}^{t} B_s$.

Given two random variables A and B, we say A is *stochastically smaller than* B, denoted by $A \preceq B$, if and only if

$$\mathbb{P}\{A \geq r\} \leq \mathbb{P}\{B \geq r\} \qquad \text{for all } r \in \mathbb{R},$$

where $\mathbb{R}$ is the set of real numbers. The random variable W_t is stochastically larger than $|S_t|$, as there is a "trimming" effect at each hop. For example, the number of leaves between z_1 and y' has a distribution similar to $\lfloor W_1 \rfloor$. But some of these leaves might not belong to S_1.

Lemma 1. *For all $t \geq 1$, we have $|S_t| \preceq W_t$.*

Lemma 2. *For all $t \in \mathbb{N}$, we have:*

(i) $$\sup_{x_1,\ldots,x_n} \sup_x \sup_y \mathbb{P}\{T_{xy} \geq t\} \leq \mathbb{P}\{nB_1 \ldots B_t \geq 1\},$$

(ii) $$\sup_{x_1,\ldots,x_n} \sup_x \mathbb{P}\left\{\sup_y T_{xy} \geq t\right\} \leq n \times \mathbb{P}\{nB_1 \ldots B_t \geq 1\},$$

(iii) $$\sup_{x_1,\ldots,x_n} \mathbb{P}\left\{\sup_x \sup_y T_{xy} \geq t\right\} \leq n^2 \times \mathbb{P}\{nB_1 \ldots B_t \geq 1\} \quad .$$

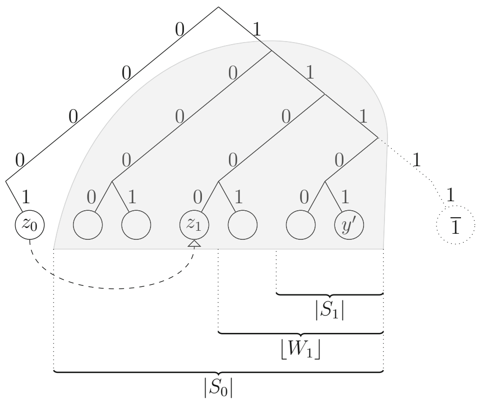

Fig. 3. The "trimming" effect

A beta random variable $B(a, b)$ has probability density function

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\, x^{a-1}(1-x)^{b-1}, \quad 0 \le x \le 1,$$

where $\Gamma(z)$ is the gamma function $\Gamma(z) = \int_0^\infty e^{-t}t^{z-1}\,\mathrm{d}t$. In order statistics theory, a basic result [18, chap.2.3] is that the r-th smallest of m i.i.d. uniform random variables has beta distribution $B(r, m+1-r)$. Plugging in $r = 1, m = k$, we have $B_t \overset{\mathcal{L}}{=} B(k) \overset{\mathcal{L}}{=} B(1, k)$ for all $t \in \mathbb{N}$. (For more about beta distribution, see [19, chap.25].) It is easy to check that for all $r > 0$ and $t \in \mathbb{N}$, we have

$$\mathbb{E}\left[(B_1 \cdots B_t)^r\right] = \mathbb{E}\left[B_1^r\right]^t = \mathbb{E}\left[B(1,k)^r\right]^t = \left(\frac{k!}{\prod_{i=1}^k (r+i)}\right)^t . \tag{1}$$

By applying this moment bound, we have the following theorem:

Theorem 3. *There exist constants c_k, c'_k and c^*_k such that:*
(i) for all $c > c_k$,

$$\sup_{x_1,\ldots,x_n} \sup_x \sup_y \mathbb{P}\left\{T_{xy} \ge c \log n\right\} \to 0 \qquad \text{as } n \to \infty;$$

(ii) for all $c > c'_k$,

$$\sup_{x_1,\ldots,x_n} \sup_x \mathbb{P}\left\{\sup_y T_{xy} \ge c \log n\right\} \to 0 \qquad \text{as } n \to \infty;$$

(iii) for all $c > c^*_k$,

$$\sup_{x_1,\ldots,x_n} \mathbb{P}\left\{\sup_{x,y} T_{xy} \ge c \log n\right\} \to 0 \qquad \text{as } n \to \infty\ .$$

In particular, we have $c_k = 1/H_k$ where H_k is the k-th harmonic number.

It is easy to check that $c'_1 = e$, since $(r+1)/\log(1+r)$ takes minimum value e when $r = e-1$. But unlike c_k, we do not have closed forms for c'_k and c^*_k. Table 1 shows the numerical values of c_k, c'_k, c^*_k for $k = 1, \ldots, 10$.

Table 1. The numerical values of c_k, c'_k, c^*_k for $k = 1, \ldots, 10$

k	c_k	c'_k	c^*_k
1	1	2.718281828	3.591121477
2	0.6666666667	1.673805050	2.170961287
3	0.5454545455	1.302556173	1.668389781
4	0.4800000000	1.105969343	1.403318015
5	0.4379562044	0.9817977138	1.236481558
6	0.4081632653	0.8950813294	1.120340102
7	0.3856749311	0.8304602569	1.034040176
8	0.3679369251	0.7800681679	0.9669189101
9	0.3534857624	0.7394331755	0.9129238915
10	0.3414171521	0.7058123636	0.8683482160

Lemma 3. *We have*

$$\lim_{k\to\infty} c_k \log k = \lim_{k\to\infty} c'_k \log k = \lim_{k\to\infty} c^*_k \log k = 1 \ .$$

Remark 1. We are not providing precise inequalities with matching lower bounds. This can be done, but in that case, one could have to distinguish between many choices for d. We have already noted that $d \geq \log_2 n$. We always have

$$T_{xy} \leq d \ .$$

Therefore, for d very large, there is a danger of having routing times that are super-logarithmic in n. Our analysis shows that this is not the case. However, the precise behavior of T_{xy}, uniformly over all x,y and d, requires additional analysis. The behavior for d near $\log_2 n$, $d = \Theta(\log n)$, and $d/\log n \to \infty$ is quite different.

Remark 2. The performance bounds of this section are of the form $c_k \log n$ with $c_k = 1/H_k$. Although formulated for fixed k, they remain valid if k is allowed to depend upon n. For example, if $k = \log n$—that is, the routing table size grows as $d \times \log n$—the expected routing time is bounded by

$$(1+o(1))\frac{\log n}{\log k} \sim (1+o(1))\frac{\log n}{\log\log n} \ .$$

Even more important is the possibility of having $O(1)$ routing time. With $k \sim n^\theta$ for $\theta \in (0,1)$, the routing table size for one computer grows as $d \times n^\theta$, and

$$\mathbb{E}\left[\sup_x \sup_y T_{xy}\right] \leq \frac{1}{\theta} + o(1) \ .$$

The parameter θ can be tweaked to obtain an acceptable compromise between storage and routing time.

4 The Random ID Model

In this section, we assume that $X_1, \ldots, X_n$ are chosen uniformly at random from $\{0,1\}^d$ without replacement. Recall that $\widehat{X}_1$ denotes the ID that is farthest from X_1. We sketch the proof that the distribution of $\widehat{T} \overset{\text{def}}{=} T_{X_1\widehat{X}_1}$ has concentrated mass.

Since rotating the hypercube does not change the distribution of the routing time, we can always assume that $X_1 = \bar{0}$ and $\widehat{X}_1 = \bar{1}$, where $\bar{b}$ denotes the all-b vector $(b, b, \ldots, b)$.

Write the routing path $\rho_{X_1\widehat{X}_1} = (z_0, z_1, \ldots)$. Let a_t be the lowest common ancestor of z_t and y', i.e., the rightmost leaf in the trie and the destination of routing. Then the sequence $(a_0, a_1, \ldots)$ can be seen as travelling downwards along the path from the root to y', i.e., the rightmost branch of the trie, with the distance of each hop being random.

This sequence can be defined equivalently as follows. Let a_0 be the root of the ID trie. Let $z_0 = \bar{0}$. From the right subtree of node a_t, select up to k paths to the bottom uniformly at random without replacement. (This is equivalent to the choice of k nodes to fill one k-bucket of z_t, the t-th hop in the search.) If that right subtree is empty, then the search terminates at a_t. Let z_{t+1} be the leaf corresponding to the rightmost one of these selected paths. Let a_{t+1} be the lowest common ancestor of z_{t+1} and y'. Let L_{t+1} be the distance from a_0 to a_{t+1}. Let $R_{t+1} = L_{t+1} - L_t$, i.e., the distance of the $(t+1)$-th hop, which is also the distance between a_t and a_{t+1}. Note that $\widehat{T} = t$ if and only if $\sum_{i=1}^{t} R_i = d$. Therefore, we can bound $\widehat{T}$ by studying the properties of $(R_t)_{t\geq 1}$.

Now instead of the ID trie of depth d, consider a full binary trie with infinite depth. Definite $(a'_0, a'_1, \ldots)$, the counterpart of $(a_0, a_1, \ldots)$ in this infinite trie, as follows. Let a'_0 be the root of the infinite trie. From the right subtree of node a'_t, select exactly k infinite downwards paths uniformly at random. (Since the probability of selecting the same path more than once is zero, "without replacement" is not necessary anymore.) Let z'_{t+1} be the rightmost of these selected paths. Let a'_{t+1} be the lowest common node of z'_{t+1} and $\bar{1}$. Let L'_{t+1} be the distance from a'_0 to a'_{t+1}. Let $G_{t+1} = L'_{t+1} - L'_t$, i.e., the distance of $(t+1)$-th hop, which is also the distance between a'_t and a'_{t+1}.

When n is large (and thus d is large), the behavior of $(R_t)_{t\geq 1}$ and $(G_t)_{t\geq 1}$ are very similar. But it is easy to see that, since each subtree of the infinite trie has exactly the same structure, $(G_t)_{t\geq 1}$ is a sequence of i.i.d. random variables with distribution

$$\mathbb{P}\{G_1 \leq i\} = \left(1 - \frac{1}{2^i}\right)^k \qquad \text{for all } i \in \mathbb{N}. \tag{2}$$

In other words, $(G_t)_{t\geq 1}$ is much easier to analyze. (Note that when $k = 1$, G_1 is simply the geometric distribution.) And it is possible to couple the random variables G_t and R_t.

When the downwards travel reaches the depth of $\log_2 n$, the routing can not last much longer. In fact, in the ID trie, a subtree whose root has depth at least

$\log_2 n$ has only $o(\log n)$ leaves with high probability [17]. Therefore, we define

$$T_n = \min \left\{ t : t \geq 1, \sum_{i=1}^{t} G_i \geq \log_2 n \right\}.$$

Lemma 4. *We have*

$$\mathbb{E}\left[\frac{T_n}{\log n}\right] \to \frac{1}{\log(2) \times \mathbb{E}\left[G_1\right]},$$

as $n \to \infty$*, and also*

$$\frac{T_n}{\log n} \to \frac{1}{\log(2) \times \mathbb{E}\left[G_1\right]} \qquad \textit{in probability},$$

as $n \to \infty$*.*

Then, by coupling, we can show Lemma 5:

Lemma 5. *We have*

$$\frac{\widehat{T} - T_n}{\log n} \to 0 \qquad \textit{in probability},$$

as $n \to \infty$*.*

We omit the proof of these two lemmas due to space limitations. For details, see [17]. For other proofs, see the appendix of this paper at `http://xingshicai.ca/kad.pdf` .

References

1. Davis, C., Fernandez, J., Neville, S., McHugh, J.: Sybil attacks as a mitigation strategy against the storm botnet. In: Proceedings of the 3rd International Conference on Malicious and Unwanted Software, Malware 2008, pp. 32–40 (2008)
2. Davis, C.R., Neville, S., Fernandez, J.M., Robert, J.-M., McHugh, J.: Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures? In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 461–480. Springer, Heidelberg (2008)
3. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
4. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: Proceedings of 1st International Conference on Peer-to-Peer Computing, pp. 101–102 (2001)
5. Steinmetz, R., Wehrle, K. (eds.): Peer-to-Peer Systems and Applications. LNCS, vol. 3485. Springer, Heidelberg (2005)
6. Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Looking up data in P2P systems. Communications of the ACM 46(2), 43–48 (2003)

7. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
8. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. SIGCOMM Computer Communication Review 31(4), 161–172 (2001)
9. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM Computer Communication Review 31(4), 149–160 (2001)
10. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 22, 41–53 (2004)
11. Crosby, S.A., Wallach, D.S.: An analysis of BitTorrent's two Kademlia-based DHTs. Rice University, Houston, TX, USA (2007)
12. Steiner, M., En-Najjary, T., Biersack, E.W.: A global view of Kad. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, pp. 117–122. ACM, New York (2007)
13. Falkner, J., Piatek, M., John, J.P., Krishnamurthy, A., Anderson, T.: Profiling a million user DHT. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, pp. 129–134. ACM, New York (2007)
14. Steiner, M., En-Najjary, T., Biersack, E.W.: Exploiting Kad: possible uses and misuses. SIGCOMM Computer Communication Review 37(5), 65–70 (2007)
15. Fredkin, E.: Trie memory. Communications of the ACM 3(9), 490–499 (1960)
16. Szpankowski, W.: Average Case Analysis of Algorithms on Sequences. Wiley, Chichester (2011)
17. Cai, X.S.: A probabilistic analysis of kademlia networks. Master's thesis, McGill University (August 2012)
18. David, H., Nagaraja, H.: Order Statistics. Wiley, Hoboken (2003)
19. Johnson, N., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, vol. 2. Wiley, Hoboken (1995)