Choosing a Random Peer

Valerie King, Department of Computer Science, University of Victoria
Jared Saia, Department of Computer Science, University of New Mexico

Abstract

We present the first fully distributed algorithm which chooses a peer uniformly at random from the set of all peers in a distributed hash table (DHT). Our algorithm has latency $O(\log n)$ and sends $O(\log n)$ messages in expectation for a DHT like Chord [17]. Our motivation for studying this problem is threefold: to enable data collection by statistically rigorous sampling methods; to provide support for randomized, distributed algorithms over peer-to-peer networks; and to support the creation and maintenance of random links, and thereby offer a simple means of improving fault-tolerance.

1 Introduction

In this paper, we address the problem of choosing a peer uniformly at random from the set of all peers in a Distributed Hash Table (DHT). Random sampling is a fundamental statistical operation; a function which chooses a random peer can be used for many types of applications, including the following:

- *Data Collection:* By randomly sampling peers, we can quickly collect the following types of useful information: peer opinions, e.g., on popular content; physical properties of network nodes, e.g., for measurement studies like [16, 15]; and environmental data, e.g., for sensor networks.
- *Supporting Randomized Algorithms:* We know of two randomized algorithms for P2P systems which require a function for choosing a random peer. The first algorithm ensures good load-balancing of computational tasks across the peers in a network [6]. The second algorithm provides a scalable solution to the Byzantine agreement problem [8]. Both algorithms critically rely on the existence of a function for choosing a random peer, but unfortunately, both results only suggest heuristics to approximate such a function.
- *Create Random Links:* Consider a network where every node has a small number of links to other random nodes. Such a network is known to be robust in the sense that it will stay well-connected even in the face of a sudden, massive number of adversarial node deletions [11]. A function for choosing a random peer allows for simple creation and maintenance

of random links, and these random links provide an extra measure of robustness.

In this paper, we will use the following DHT model [17]. We will assume that the "key space" of the DHT is scaled so it is in the range $(0, 1]$ and will think of the DHT as a circle with unit circumference, which we will call the *unit circle*. We assume that n peers are connected in the DHT and that all of the n peers are mapped to locations on the unit circle which we call *peer points*. We assume that the n peer points are distributed uniformly at random on the unit circle[1]. We further assume that the DHT provides two basic operations: h and *next*. For a point x on the unit circle, $h(x)$ is the peer whose peer point is closest in clockwise distance to x. For a given peer p, $next(p)$ returns the peer whose peer point is closest in clockwise distance to p's peer point.

Our problem then is to design a scalable, distributed function which chooses a peer uniformly at random from the set of all peers in the DHT. We want this function to use only the basic DHT operations h and *next* and we want it to be scalable in the sense that latency and bandwidth will be at most polylogarithmic in n.

A simple heuristic for this problem is to choose a random point x on the unit circle and return $h(x)$. Unfortunately, this heuristic, although simple, is biased. The probability that a peer p is chosen by this heuristic is proportional to the length of the arc between the peer point for p and the closest counter-clockwise peer point. The lengths of these arcs vary widely. In particular, with high probability, the longest arc is of length $\Theta(\log n/n)$ [17] and the shortest arc is of length $\Theta(1/n^2)$ (see Theorem 11). Thus, the peer with the longest arc will be chosen $\Theta(n \log n)$ times more frequently than the peer with the shortest arc. To remove this bias, we require a more sophisticated algorithm.

1.1 Our Results

Our main result is stated in Theorems 9 and 10, which are summarized here. These theorems show that, with high probability, our algorithm:

- always chooses each peer p with probability *exactly* $1/n$;
- has expected latency $O(\log n)$ and sends $O(\log n)$ messages in expectation.

In particular, for any base hash function of the DHT, with probability $1 - 3/n$, our algorithm has these two properties every time it is called by any peer in the DHT. The expected latency and message costs assume that the DHT computes the function h with $O(\log n)$ latency and $O(\log n)$ messages, and that it computes the function *next* with $O(1)$ latency and $O(1)$ messages.

1.2 Related Work

Gkansidis et. al. addresses the problem of choosing a random peer in a P2P system [4]. They show that random walks can provide a good approximation

[1] As is standard, we use the random oracle model [2] for the base hash function of the DHT.

to uniform sampling for networks where the gap between the first and second eigenvalues of the transition matrix is constant. Unfortunately, their result only approximates uniform sampling and the closeness of the approximation is impossible to formally state without knowledge of the second eigenvalue of the network. See also Law and Siu [7] who also use random walks to approximately sample peers.

There are several results on adding load-balancing extensions to the basic DHT model. These results seek to more equitably map the function h across the peers. One technique is that of *virtual nodes*(see e.g. [17]): each peer maps to $O(\log n)$ peer points on the unit circle. A peer is then responsible for all points which are closest in clockwise distance to any of its $O(\log n)$ peer points. While virtual nodes do improve load-balancing, one drawback, as noted in [3] and [5], is that they also increase the bandwidth required for basic network maintenance. There are several load-balancing techniques which do not use virtual nodes [3, 1, 14, 5]. Generally these techniques work by dynamically "reassigning" hash space among the peers to ensure that no peer is ever responsible for too large a portion.

We have assumed a standard DHT which has no load-balancing extensions. We make this assumption for two reasons. First, we would like our protocol to be applicable for a wide-range of DHTs. Unfortunately, there is currently no consensus about the best way to add load-balancing extensions to a DHT so assuming some particular method would hurt generality. We believe that adding load-balancing extensions to the DHT generally makes the problem of choosing a random peer easier and so the results we have for the basic DHT can be easily adapted to a DHT which has load-balancing extension. Second, we want our results to hold even in the presence of malicious faults and we are not aware of any DHTs with load-balancing extensions which are provably robust to malicious faults.

1.3 Notation

For any two points x and y on the unit circle, we let $d(x, y)$ be the distance from x to y traveling clockwise along the unit circle (i.e. if $y \geq x$, then $d(x, y) = y - x$ else $d(x, y) = (1 - x) + y$). Let $num(x, y)$ denote the number of peer points in the half-closed interval $(x, y]$ traveling clockwise from x to y along the unit circle.

For a given peer p, we let $l(p)$ be p's peer point. We note that k applications of $next$ returns the k^{th} next peer in the clockwise ordering around the circle from $l(p)$ and is denoted $next^{(k)}$. We assume that a single application of $next$ has $O(1)$ latency and requires $O(1)$ messages to be sent. We assume that computing $h(x)$ for some arbitrary point x has t_h latency and requires m_h messages to be sent. Typically $t_h = O(\log n)$ and $m_h = O(\log n)$.

The rest of this paper is laid out as follows. In Section 2 we give an algorithm which allows a peer to estimate n to within a constant factor. We then use this algorithm in Section 3 when we present our algorithm for choosing a peer uniformly at random. We conclude and give directions for future work in Section 4.

2 Estimating n

In this section, we describe an algorithm which allows a peer p to estimate n to within a constant multiplicative factor. Our technique is similar to that of Manku in [10]. The algorithm has two steps. The first is to estimate n to within a constant exponent, by measuring the distance between two consecutive peer points and taking its inverse. The second uses the first estimate $\hat{n_1}$ to estimate $\ln n$ within a constant factor. It then finds the length t of the interval which contains $s = c_1 \ln \hat{n_1}$ peer points and uses the ratio of s over t to get the final estimate $\hat{n_2}$ of n. The constant c_1 determines the tightness of the estimate.

The algorithm is given below.

Estimate n

1. $\hat{n_1} \leftarrow [d(l(p), l(next(p)))]^{-1}$;
2. $s \leftarrow c_1 \ln \hat{n_1}$;
3. $t \leftarrow d(l(p), l(next^{(s)}(p)))$;
4. Return $\hat{n_2} \leftarrow s/t$.

We will now show that the algorithm *Estimate n* returns a constant factor approximation to n with high probability. To show this, we will first need the following two lemmas. The first lemma is similar to Mahlki et. al. [9].

Lemma 1. *With probability at least $1 - 1/n$: (property 1) h has the property that for any peer, p,*

$$\ln n - \ln \ln n - 2 \leq \ln \left(\frac{1}{d(l(p), l(next(p)))} \right) \leq 3 \ln n$$

Consider some interval I of the unit circle. We say that I is *anchored* if I has a peer point, p, at its counterclockwise endpoint. We say that p is the *anchor point* for I. The proof of the following lemma is in Appendix A.

Lemma 2. *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 n sufficiently large, with probability at least $1 - 1/n$,the following (property 2) is true for h:*

- *For any anchored interval I on the unit circle, if the number of peers that I contains other than the anchor point is greater than $C\alpha_1 \ln n$ and less than $C\alpha_2 \ln n$, then I is of length between $C(1-\epsilon)\alpha_1(\ln n/n)$ and $C(1+\epsilon)\alpha_2(\ln n/n)$*

Choose Random Peer

While TRUE do :

1. $r \leftarrow$ random number in $(0, 1]$;
2. If $d(r, l(h(r))) < \lambda$ then return $h(r)$;
3. Else:
   (a) $first \leftarrow h(r)$; $T \leftarrow d(r, l(first)) - \lambda$;
   (b) Repeat $6 \ln n'$ times or until $T < 0$:
      i. $T \leftarrow T + d(l(first), l(next(first))) - \lambda$;
      ii. if $T < 0$ return $next(first)$, else $first \leftarrow next(first)$.

Figure 1: Algorithm for choosing a random peer.

For given functions $f(n)$ and $g(n)$, we say that $f(n)$ is a (γ_1, γ_2) *approximation* of $g(n)$ if $\gamma_1 g(n) \leq f(n) \leq \gamma_2 g(n)$. We can now give the main lemma of this section.

Lemma 3. *With probability at least* $1 - 2/n$, *h is such that the output of the algorithm* Estimate n, *for all peers, is a* $(2/7 - \epsilon_1, 6 + \epsilon_1)$ *approximation of* n, *for* ϵ_1 *any positive constant and* n *sufficiently large.*

Proof. Lemma 1 says that with high probability, s is a $(\beta, 3)$ approximation to $c_1 \log n$ for any fixed $\beta < 1$ when n is sufficiently large. Similarly, Lemma 2 shows that t is a $(\beta - \epsilon, 3 + \epsilon)$ approximation to $(c_1 \log n)/n$ for any $\epsilon > 0$, for n and c_1 sufficiently large. Thus, $\hat{n_2}$ is a $\left(\frac{\beta}{3+\epsilon}, \frac{3}{\beta-\epsilon}\right)$-approximation to n for c_1 and n sufficiently large. □

3 Choosing a random peer

Our algorithm for choosing a random peer is presented in Figure 1. In this algorithm, we let $\hat{n}$ be a (γ_1, γ_2)-approximation to n, γ_1, γ_2 constants. We further let $n' = \hat{n}/\gamma_1$ and let $\lambda = 1/(7n')$. Then $n' \geq n$ and is $\Theta(n)$ and $\lambda \leq 1/(7n)$ and is $\Theta(1/n)$. The main idea of the algorithm is to partition the unit circle into disjoint intervals so that for each peer there is a set of intervals assigned to that peer whose lengths sum exactly to λ.

The algorithm randomly selects a random number from $(0, 1]$. If there is a peer p such that $d(r, l(p)) < \lambda num(r, l(p))$ with $num(r, l(p)) \leq 6 \ln n'$, then the algorithm returns the first such peer. Else the algorithm repeats until a peer is returned. We will show that with high probability, the hash function has properties which imply that the expected number of repetitions of the algorithm is $O(1)$.

3.1 Proof of Correctness

We say that the algorithm *assigns* a point x on the unit circle to a peer p if the algorithm returns p when x is the random number chosen in step 1. In this section, we show that the algorithm assigns to each peer a set of disjoint intervals whose lengths sum to λ.

Lemma 4. *For any point r on the unit circle, if r is assigned by the algorithm to a peer p, then $d(r, l(p)) < \lambda num(r, l(p))$ and $num(r, l(p)) \leq 6 \ln n'$. If there is more than one such peer, then the algorithm assigns r to the closest one, i.e., the one such that $d(r, l(p))$ is minimal.*

Proof. If $r = l(p)$ or is within distance λ of $l(p)$ and p is the next peer whose peer point is clockwise from r then line 2 of the algorithm assigns r to p. In this case, $num(r, l(p)) = 1$ and $d(r, l(p)) < \lambda$.

If line 3 is executed, the algorithm visits a succession of peer points going clockwise from r. Let p_i represent the peer whose peer point is the i^{th} encountered (here, $p_1 = h(r)$). In line 3a, T is set to $d(r, l(p_1)) - \lambda$. It is easy to see by induction that at the i^{th} repetition of line 3b(ii), $T = d(x, l(p_{i+1})) - \lambda(i + 1) = d(x, l(p_{i+1})) - \lambda num(x, l(p_{i+1})$. The algorithm returns the first peer p_i such that $T < 0$, i.e., $d(r, l(p_i)) < \lambda num(x, l(p_i))$, provided that such a peer is encountered within $6 \ln n'$ repetitions.

□

For any peer p, let $Int(p) = (x, l(p)]$ be the half-closed interval on the unit circle whose endpoint x is the closest point counterclockwise from $l(p)$ such that $d(x, l(p)) \geq \lambda num(x, l(p))$.

Lemma 5. *Let p, p' be any peers such that $num(Int(p)) \leq 6 \ln n'$ and $p \neq p'$. Then:*

1. *Every point assigned by the algorithm to p lies in $Int(p)$.*
2. *Every point in $Int(p)$ is assigned by the algorithm to a peer whose peer point lies in $Int(p)$.*
3. *Either $Int(p) \subset Int(p')$, $Int(p') \subset Int(p)$, or $Int(p) \cap Int(p') = \emptyset$.*

Proof. Proof of (1): Let $Int(p) = (x, l(p)]$. Let y be a point assigned to p. Then $d(y, p) < \lambda num(y, l(p))$, by Lemma 4. Assume to the contrary that y lies outside $Int(p)$.

We first look at the case that there is no peer point in $[y, x]$. Then $d(y, p) = d(y, x) + d(x, p) \geq d(x, p) \geq \lambda num(x, p) = \lambda num(y, p)$, contradicting the assumption that the algorithm would have assigned y to p.

Alternatively, let p' be the peer whose peer point is closest to x in $[y, x]$ (or equal to x if x is a peer point). By assumption, since y was assigned to p, $d(y, l(p)) < \lambda num(y, p)$. Now, $d(y, l(p)) = d(y, l(p')) + d(l(p'), l(p))$ and $num(y, p) = num(y, l(p')) + num(l(p'), l(p))$. Hence we have

$$d(y, l(p')) + d(l(p'), l(p)) < \lambda num(y, l(p')) + \lambda num(l(p'), l(p)).$$

Since $d(l(p'), l(p)) \geq (d(x, l(p)) \geq \lambda num(x, l(p))$ and $num(l(p'), l(p)) = num(x, l(p))$, the above inequality is preserved when we subtract $d(l(p'), l(p))$ from the left-hand side and $\lambda num(l(p'), l(p))$ from the right-hand side. This implies:

$$d(y, l(p')) < \lambda num(y, l(p')).$$

By Lemma 4, the algorithm would have assigned y to p' since $d(y, l(p')) < d(y, l(p))$, contradicting our assumption.

Proof of (2): This follows from the fact that every point y in $lnt(p)$ has the property that $d(y, l(p)) < \lambda num(y, l(p)))$. Hence by Lemma 4, y is either assigned to p or some closer peer in $[y, l(p)]$.

Proof of (3): This is similar in technique to the proof of (1) and is left to the reader.

□

Lemma 6. *The set of intervals assigned to any peer p with $num(Int(p)) \leq 6 \ln n'$ has total length λ.*

Proof. The proof is by induction on the size of $num(Int(p))$ where p is any peer.

Base Case: $num = 1$. In this case, $Int(p) = (l(p) - \lambda, l(p)]$. Lemma 5 (2) implies that every point in $Int(p)$ is assigned to p and Lemma 5 (1) implies that no other point is assigned to p so the single interval assigned to p has length λ.

Induction step: Suppose $num(Int(p)) = k$. Then there are $k - 1$ peer points within $Int(p)$ excluding $l(p)$. By Lemma 5(3), each of these peer points p' have $Int(p') \subset Int(p)$. Since $Int(p')$ does not contain $l(p)$, $num(Int(p') < k$. By the induction assumption, each peer p' is assigned an interval of length λ, for a total of $(k-1)\lambda$. By Lemma 5 (2), every point in $Int(p)$ is assigned to a peer in $Int(p)$. Hence since $d(Int(p)) = \lambda k$, $k\lambda - (k-1)\lambda = \lambda$ has been assigned to p. By Lemma 5 (1), no other points on the unit circle have been assigned to p. Hence p has been assigned a set of intervals whose lengths add up to λ. □

We will show that with high probability, every peer p has $num(Int(p)) \leq 6 \ln n'$.

Lemma 7. *With probability greater than $1 - 1/n$, (property 3) h has the property that any interval containing more than $6 \ln n$ peer points has length greater than $\ln n/n$.*

Proof. We will show that no interval of length less than $(\ln n)/n$ contains more than $6 \ln n$ peer points. The analysis follows from the balls and bins paradigm. Partition the unit circle into disjoint consecutive intervals (bins) of length $(\ln n)/n$. Let X be the number of balls in any one bin. Then $E[X] = \ln n$. By the Chernoff bound, $Pr(X > (1+\delta)E[X]) < e^{-2E[X]} = 1/n^2$ for $\delta \geq 2$.

Let $\delta = 2$. With probability $1/n^2$, no consecutive pair of bins contains more than $2(1+\delta)E[X] = 6 \ln n$ peer points which implies that no interval of length $(\ln n)/n$ in the unit circle contains more than $6 \ln n$ peer points. □

Lemma 8. *Let h be a random hash function such that properties (1)–(3) hold. Then for every peer p, $num(Int(p)) \leq 6 \ln n'$.*

Proof. From Lemma 1 and Lemma 2, we know that $6 \ln n' \geq 6 \ln n$ and that $(\ln n')/n' \leq (\ln n)/n$. By Lemma 7, for any interval $(x, y]$, if $num(x, y) \geq 6 \ln n' \geq 6 \ln n$ then $d(x, y) \geq \ln n/n \geq \ln n'/n' > \lambda num(x, y)$. Hence if x is any point such that $num(x, l(p)) \geq 6 \ln n'$ then $x \notin Int(p)$. It follows that $num(Int(p)) < 6 \ln n'$.

□

Theorem 9. *Let h be a random function. Then with probability at least $1-3/n$, h has properties (1)-(3). In this case, our algorithm chooses each peer with probability $1/n$.*

Proof. Let p be any peer. From Lemma 8, $num(Int(p)) \leq 6 \ln n'$. From Lemma 6, each such peer is assigned a set of points whose lengths are equal. Since each point which is assigned to a peer has an equal chance of being chosen, the probability of choosing any peer is equal to the probability of choosing any other peer, or $1/n$. □

3.2 Latency and Bandwidth

Theorem 10. *Our algorithm has latency $O(t_h + \log n)$ and sends $O(m_h + \log n)$ messages in expectation. In particular, if $t_h = m_h = O(\log n)$, then our algorithm has latency cost $O(\log n)$ and sends $O(\log n)$ messages in expectation.*

Proof. It is easy to see that $\hat{n}$ can be computed with $O(t_h + \log n)$ latency and with $O(m_h + \log n)$ messages, since there are $O(1)$ applications of h and $O(\log n)$ applications of *next*.

We note that every iteration of the body of the while loop of the algorithm also has $O(t_h + \log n)$ latency and sends $O(m_h + \log n)$ messages, since there is one application of h and $O(\log n)$ applications of *next*.

Now we bound the expected number of times the body of the while loop must be iterated until it succeeds. Since a disjoint arc length of λ has been assigned to each peer, the probability of finding an assigned point is $n\lambda$ or $\Omega(1)$. Since each trial is independent, the number of trials needed to find an assigned point is a geometric random variable with probability $n\lambda$. In particular, the expected number of trials is no greater than $1/(n\lambda)$ or $O(1)$. This implies that the expected latency of the entire algorithm is $O(t_h + \log n)$ and the expected number of messages sent is $O(m_h + \log n)$. □

4 Conclusion and Open Problems

We have presented the first algorithm which chooses a peer uniformly at random from the set of all peers in a DHT. Numerous open problems remain including the following:

- Our algorithm is relatively simple and has small asymptotic resource costs. We would like to empiricially evaluate it to determine if it will work well in practice. Is it possible to reduce the constants in the asymptotic notation any further?

- Many peer-to-peer networks like Gnutella have much less structure than a DHT. Are there efficient algorithms to choose random peers in semi-structured peer-to-peer networks?

- In some applications, we may want to choose a peer with a biased probability. For example, we may want to choose a peer with probability that is inversely proportional to its distance from us on the unit circle. Are there efficient algorithms to choose a random peer with specifically biased probabilities?

References

[1] M. Alder, E. Halperin, R. Karp, and V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *ACM Symposium on Theory of Computing (STOC)*, 2003.

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *The First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[3] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In *Proceedings of the Second Internation Peer to Peer Symposium (IPTPS)*, 2003.

[4] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Conference of the IEEE Communications Society (INFOCOM)*, 2004.

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

[6] D. Karger and M. Ruhl. New algorithms for load balancing in peer-to-peer systems. In *Proceedings of the Fourth Internation Peer to Peer Symposium (IPTPS)*, 2004.

[7] C. Law and K.-Y. Siu. Distributed construction of random expander graphs. In *Conference of the IEEE Communications Society (INFOCOM)*, 2003.

[8] S. Lewis and J. Saia. Scalable byzantine agreement. Technical report, University of New Mexico, 2004.

[9] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly network. In *ACM Conference on the Principles of Distributed Computing (PODC)*, 2002.

[10] G. Manku Routing networks for distributed hash tables In *ACM Conference on the Principles of Distributed Computing (POD C)*, 2003.

[11] R. Motwani and P. Raghavan. *Randomized Algorithms*, chapter 5.3. Cambridge University Press, 1995.

[12] R. Motwani and P. Raghavan. *Randomized Algorithms*, chapter 4. Cambridge University Press, 1995.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[14] M. Naor and U. Weider. Novel architectures for p2p applications: the continuous-discrete approach. In *SPAA*, 2003.

[15] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[16] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking*, 2002.

[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM 2001 Technical Conference*, San Diego, CA, USA, August 2001.

A Proofs

In this section, we present proofs of some of the theorems stated in the paper.

Theorem 11. *With high probability, when n peers are distributed uniformly at random on the unit circle, the shortest arc length between two peers is* $\Theta(1/n^2)$

Proof. First we show that the smallest arc length is $O(1/n^2)$. Imagine that we are tossing the peers onto the unit circle one at a time. Let k be such that $0 \leq k < 1$ and let χ_i be the event that no arc of size k or less is formed up to the time the i-th peer is thrown in. Then we can say that:

$$P(\chi_i|\chi_{i-1}) \leq 1 - k(i-1).$$

Thus we have:

$$
\begin{aligned}
P(\chi_n) &= \prod_{i=2}^{n} P(\chi_i|\chi_{i-1}); \\
&\leq \prod_{i=1}^{n}(1-ki); \\
&\leq e^{-\sum_{i=1}^{n} ki}; \\
&\leq e^{-k(n+1)n/2}.
\end{aligned}
$$

We note that the last term is a constant only when $k = O(1/n^2)$

To see that the smallest arc length is $\Omega(1/n^2)$, let $\bar{\chi_n}$ be the probability that some arc is of length k or less when the n-th peer is thrown. Then note that:

$$
\begin{aligned}
P(\bar{\chi_n}) &= 1 - P(\chi_n); \\
&\geq 1 - e^{-k(n+1)n/2}.
\end{aligned}
$$

For $k = \Omega(1/n^2)$, this probability can be made arbitrarily close to 1. □

We now present the proof of Lemma 2 which we repeat here for convenience.

Lemma 2: Let $\alpha_1, \alpha_2, \epsilon$ and k be fixed positive constants with $\alpha_1 < \alpha_2$ and $0 \leq \epsilon \leq 1$. Let C be a positive constant depending only on $\alpha_1, \alpha_2, \epsilon$ and k. Then for n sufficiently large, with probability $1 - n^{-k}$,the following statement is true:

- *For any anchored interval I on the unit circle, if the number of peers that I contains other than the anchor point is greater than $C\alpha_1 \ln n$ and less than $C\alpha_2 \ln n$, then I is of length between $C(1-\epsilon)\alpha_1(\ln n/n)$ and $C(1+\epsilon)\alpha_2(\ln n/n)$*

Proof. Fix some peer point p on the unit circle. Let I_s be the interval starting at p and extending clockwise for a distance of $C(1-\epsilon)\alpha_1(\ln n/n)$. Let I_l be the interval starting at p and extending clockwise for a distance of $C(1+\epsilon)\alpha_2(\ln n/n)$. We will now show that with high probability, I_s contains less than or equal to $C\alpha_1 \log n$ other peer points and I_l contains greater than or equal to $C\alpha_2 \ln n$ other peer points.

Let X_s be a random variable giving the number of peer points other than p that fall in the interval I_s. Note that a single peer point falls in the interval I_s with probability $C(1-\epsilon)\alpha_1(\ln n)/n$, so by linearity of expectation:

$$E(X_s) = C(1-\epsilon)\alpha_1(n-1)(\ln n)/n.$$

Chernoff bounds [12] tell us that for any δ, $0 \leq \delta \leq 1$:

$$P(X_s > (1+\delta)E(X_s)) \quad < \quad e^{-\frac{\delta^2 E(X_s)}{3}};$$

Setting $\delta = \epsilon/(1-\epsilon)$, ensures that

$$\begin{aligned}(1+\delta)E(X_s) &= C\alpha_1(\ln n)(n-1)/n \\ &\leq C\alpha_1 \ln n\end{aligned}$$

Thus:

$$\begin{aligned}P(X_s > C\alpha_1 \ln n) &< e^{-\frac{C\epsilon^2\alpha_1(n-1)\ln n}{3n(1-\epsilon)}}. \\ &\leq e^{-\frac{C\epsilon^2\alpha_1 \ln n}{6(1-\epsilon)}}. \\ &\leq e^{-\frac{C\epsilon^2\alpha_1 \ln n}{6}}.\end{aligned}$$

Where the second line follows if we assume that $n \geq 2$ (since then $(n-1)/n \geq 1/2$) and the third line follows by our assumption that $0 \leq \epsilon \leq 1$.

Now let X_l be a random variable giving the number of peer points other than p that fall in the interval I_l. A single peer point falls in the interval X_l with probability $C(1+\epsilon)\alpha_2(\ln n/n)$ so by linearity of expectation:

$$E(X_l) = C(1+\epsilon)\alpha_2(n-1)(\ln n)/n.$$

Chernoff bounds [13] tell us that for any δ, $0 \leq \delta \leq 1$:

$$P(X_l < (1-\delta)E(X_l)) \quad < \quad e^{-\frac{\delta^2 E(X_l)}{2}};$$

We want to choose δ such that $(1-\delta)E(X_l) \geq C\alpha_2 \ln n$. Assume that $(n-1)/n \geq \gamma$ for some value $\gamma < 1$. Then we know that $E(X_l) \geq C\gamma(1+\epsilon)\alpha_2 \ln n$. Thus to ensure that $(1-\delta)E(X_l) \geq C\alpha_2 \ln n$, it suffices if $(1-\delta) \geq \frac{1}{\gamma(1+\epsilon)}$. In other words, we need $\delta \leq 1 - \frac{1}{\gamma(1+\epsilon)}$. To use Chernoff bounds, we have the additional constraint that $0 \leq \delta \leq 1$. Thus, it must be the case that $\frac{1}{\gamma(1+\epsilon)} < 1$. Choosing $\gamma = \frac{1+\epsilon/2}{1+\epsilon}$ satisfies all of these constraints and requires that $\delta \leq \frac{\epsilon}{2+\epsilon}$. To recap, the key assumption we are making is that $(n-1)/n \geq \frac{1+\epsilon/2}{1+\epsilon}$ which is true when $n \geq \frac{2(1+\epsilon)}{\epsilon}$.

Setting $\delta = \frac{\epsilon}{2+\epsilon}$ (and assuming that $n > \frac{2(1+\epsilon)}{\epsilon}$), we have that.

$$\begin{aligned}P(X_l < C\alpha_2 \ln n) &< e^{-\frac{C\alpha_2(n-1)\epsilon^2(1+\epsilon)\ln n}{2n(2+\epsilon)^2}}. \\ &\leq e^{-\frac{C\alpha_2\epsilon^2(1+\epsilon)\ln n}{4(2+\epsilon)^2}}. \\ &\leq e^{-\frac{C\alpha_2\epsilon^2 \ln n}{36}}.\end{aligned}$$

Where the second line follows if we assume that $n \geq 2$ (since then $(n-1)/n \geq 1/2$) and the third line follows by our assumption that $0 \leq \epsilon \leq 1$.

Now for the peer p, consider any anchored interval I that has p as its anchor point. Say that I is *small* if it has length less than or equal to $C(1-\epsilon)\alpha_1(\log n/n)$, and *large* if it has length greater than or equal to $C(1+\epsilon)\alpha_2(\log n/n)$. The bad event for p is that either 1) I is small and I contains greater than $C\alpha_1 \log n$ peer points other than p or 2) I is large and I contains less than $C\alpha_2 \log n$ peer points other than p. Let ξ_p be the bad event for the peer p. Then, by a simple union bound, we can say that

$$\begin{aligned} P(\xi_p) &\leq e^{-\frac{C\epsilon^2\alpha_1 \ln n}{6}} + e^{-\frac{C\alpha_2\epsilon^2 \ln n}{36}} \\ &\leq 2e^{-\frac{C\epsilon^2\alpha_1 \ln n}{36}} \end{aligned}$$

Now let ξ be the event that for *any* peer p, there exists an interval I anchored at p such that 1) I is small and I contains greater than $C\alpha_1 \ln n$ peer points other than p or 2) I is large and I contains less than $C\alpha_2 \ln n$ peer points other than p. In other words, ξ is the event that the statement of the lemma fails. Again by a simple union bound, we can say that:

$$P(\xi) \leq 2ne^{-\frac{C\epsilon^2\alpha_1 \ln n}{36}}$$

The last equation can be made arbitrarily small for C chosen large enough. $\square$