The Expansion and Mixing Time of Skip Graphs with Applications

James Aspnes[*†] Udi Wieder [‡§]

June 11, 2008

Abstract

We prove that with high probability a skip graph contains a 4-regular expander as a subgraph and estimate the quality of the expansion via simulations. As a consequence, skip graphs contain a large connected component even after an adversarial deletion of nodes. We show how the expansion property can be used to sample a node in the skip graph in a highly efficient manner. We also show that the expansion property can be used to load balance the skip graph quickly. Finally, it is shown that the skip graph could serve as an unstructured P2P system, making it a good candidate for a hybrid P2P system.

1 Introduction

Skip graphs [3] and SkipNets [7] are randomized distributed data structures designed for use in peer-to-peer (P2P) storage systems. Like Distributed Hash Tables (DHTs), skip graphs scale gracefully, and offer excellent query complexity [14]. Skip graphs have an advantage over DHTs in that they directly support *range queries*, while DHTs provide exact search only. Much of the usefulness of skip graphs depends on their properties as random graphs. It was previously shown [3] that (with high probability) skip graphs have expansion ratio $\Omega(1/\log n)$: every subset of $m \leq n/2$ nodes of a skip graph has $\Omega(m/\log n)$ neighbors. This bound is surprisingly low given that skip graphs have average degree $O(\log n)$, but experimental examination of small cases suggested it was the best possible.

In this paper we prove that with high probability a skip graph has an expansion ratio of $\Omega(1)$: every subset of $m \leq n/2$ nodes has $\Omega(m)$ neighbors. In fact, we prove a much stronger result: with high probability, a skip graph contains a degree-4 regular expander as a subgraph; i.e., it contains a degree-4 regular subgraph with expansion ratio $\Omega(1)$. The edges of this expander for each node can be computed using only local information in $O(1)$ expected time and $O(\log n)$ time with high probability. Consequences of the embedded expander include:

- **Fault tolerance:** The expansion property is equivalent to the property that a deletion of k nodes may isolate from the primary component at most $O(k)$ nodes. In other words, even if

[*]Supported in part by NSF grants CCR-0098078, CNS-0305258, and CNS-0435201.
[†]Department of Computer Science, Yale University. New Haven, CT 06520, USA. Email: aspnes@cs.yale.edu.
[‡]Microsoft Research Silicon Valley. Email: uwieder@microsoft.com.
[§]Research done while at the Weizmann Institute of Science.

a constant fraction of the nodes are deleted by an *adversary*, still a constant fraction of the nodes would remain connected in a single component.

- **Efficient sampling:** A random walk in an expander graph quickly converges towards the stationary distribution. This could be used in order to sample uniformly a random node. We present several sampling algorithms and show that they are much faster than the currently best known algorithms.
- **Low hitting times:** Random walks on expanders have the property of hitting a large set of nodes fast and with high probability. This can be used for a variety of applications such as load balancing, gathering statistics on the nodes of the skip graph and for finding highly replicated data items. It is known that in unstructured P2P systems, when the underlying graph expands there are algorithms for locating data items which are more efficient than simple flooding [6]. Furthermore, it was shown [11] that in some cases unstructured P2P systems can locate items faster than structured systems. The skip graph therefore is shown to be competitive with *unstructured* P2P systems, thus making it an excellent candidate for a hybrid P2P system.

1.1 Comparison with previous work

Expanding networks The advantages of an expanding topology are well known, the literature is abundant with variations of expanding networks. In the context of dynamic P2P networks we are aware of only two previous approaches. Naor and Wieder [16] build an overlay network that emulates the Margulis [15] explicit construction of expanders. The quality of the expansion property depends upon the load balancing of the i.d. selection scheme. The expanding network itself does not support a lookup functionality and assumes the existence of some external lookup. The main advantage of the construction in [16] is its guaranteed expansion. Its main drawback compared to the present work is that it has a rather large overhead in maintaining the network and keeping the i.d. selection well balanced, thus making it an appealing theoretical solution yet somewhat unpractical.

The second suggestion for an expanding overlay network was made by Law and Siu [10]. They suggest building d random Hamiltonian cycles, which have an optimal spectral gap w.h.p. and are thus expanding [5]. The main advantages of this scheme are its relative simplicity and its optimality in the sense that w.h.p. the graph will have the (almost) largest possible spectral gap with respect to the degree. The basic construction does not support a lookup operation. A more elaborate construction which does support lookup needs a logarithmic degree and then a lookup takes $O(\log n)$ hops on expectation. Asymptotically, this is almost in par with skip graphs which can perform a lookup in $O(\log n/\log\log n)$ hops using the Neighbor of Neighbor approach. While Law and Siu do not provide simulation results, it seems that the lookup operation would have a relatively high constant hidden in the O-notation. Their construction also needs a sampling protocol as a primitive and suggests using random walks in order to obtain the samples. But the uniformity of the distribution produced by the random walk depends on the expansion, while the expansion depends on the uniformity. This mutual dependency means that an error in the early stages of the construction may accumulate and ruin the expanding property. In contrast, the expansion in our construction depends upon the random bits generated independently by each node separately, so there is no mutual dependency between the correctness of the join algorithm and the expansion. Furthermore, it should be noted that Law and Siu's construction can easily be implemented on top of the skip graph using the sampling algorithm of Section 3, obtaining the best properties of both constructions. Indeed this is implicitly done by the method of Zatloukal and Harvey [19], which builds a variant of a skip graph incorporating two random Hamiltonian cycles.

Sampling schemes In the context of P2P systems, a random walk sampling scheme was previously suggested by Law and Siu [10] (as mentioned above) and by Gkantsidis *et al* [6]. Obtaining good samples using such random walks requires *a priori* knowledge of the spectral gap of the graph. Our simulations imply that the spectral gap of the skip graph is well concentrated and therefore can be known in advance, so that random walks work well in a skip graph. See Section 3.2.

A different sampling scheme for DHTs was suggested by King and Saia [9]. Their scheme yields an *exact* uniform distribution and runs in expected logarithmic time. Recent work by King *et al* [8] claims that the expected running time is at least $11 \log n$. Empirical testing shows that our algorithm runs much faster, albeit at the cost of slight deviations from uniformity. A running time of about $2 \log n$ produces a sample from a distribution that is close enough to uniform for most conceivable applications (see Section 3.2).

1.2 A brief review of skip graphs

In a skip graph, each node represents a resource to be searched. Node x holds two fields: the first is a key, which is arbitrary and may be the resource name. Nodes are ordered according to their keys. We assume for notational convenience that the keys are the integers $1, 2, \ldots, n$. Since the keys have no function in the construction other than to provide an ordering and a target for searches there is no loss of generality. The second field is a membership vector $m(x)$ which is for convenience treated as an infinite string of random bits chosen independently by each node. In practice, it is enough to generate an $O(\log n)$-bit prefix of this string with overwhelming probability.

The nodes are ordered lexicographically by their keys in a circular doubly-linked list S_ϵ so that node i is connected to $i - 1 \mod n$ and $i + 1 \mod n$. For each finite bit-vector σ, an additional circular doubly-linked list S_σ is constructed by taking all nodes whose membership vectors have σ as a prefix, and linking adjacent nodes in the lexicographic key order. More formally, let $m(x) \upharpoonright k$ be the restriction of $m(x)$ to its first k bits; then nodes $x < y$ are connected by an edge if there exists some k such that $m(x) \upharpoonright k = m(y) \upharpoonright k$, and either (a) $m(z) \upharpoonright k \neq m(x) \upharpoonright k$ for each z between x and y, or (b) $m(z) \upharpoonright k \neq m(x) \upharpoonright k$ for all $z > x$ and all $z < y$. In analyzing a skip graph as a graph, we treat each pair of links as a single undirected edge, and take the union of the resulting edge sets for all lists S_σ.

It is shown in [3, 7] that a new node (with a new key) can join the system in $O(\log n)$ time sending $O(\log n)$ messages. It is also shown [3, 7] that every node can locate (the node assigned to) every key in $O(\log n)$ hops. Manku *et al* [14] showed that path lengths could be reduced to $O(\log n / \log \log n)$ using the Neighbor of Neighbor approach.

The main merit of skip-graphs is the following: edges do not depend on the keys themselves but rather on their ordering and the random vectors. Thus the keys may be arbitrary and can carry *semantic meaning*. Furthermore, since the nodes are ordered by their keys, the skip-graph data structure supports *prefix search*. This is in stark contrast with the other networks we discuss, which require that keys be random.

2 Main result

We will show that skip graphs have an edge expansion of some small $\epsilon > 0$ with high probability. Throughout the paper let G denote a skip graph of n vertices. For a vertex set U define $\delta(U)$ to be the number of edges with exactly one endpoint in U.

Theorem 2.1. *With high probability*[1] *the following event occurs: G has a subgraph G' of degree 4 such that for every set $U \subset V$ of size at most $\frac{n}{2}$ it holds that $|\delta(U)| \geq \epsilon|U|$ (where $\epsilon > 0$ is independent of n). The probability is taken over the choices of membership vectors.*

Remark: We do not state what is the largest ϵ for which Theorem 2.1 is correct. Indeed, in the proof we are liberal in making ϵ as small as necessary. An estimate of the expansion is obtained via simulations (see Section 3.2). Since we deal with a bounded degree graph, edge expansion and node expansion are equivalent up to constant factors.

The subgraph G' is a union of two $2-$regular subgraphs. In a skip graph, all the nodes are connected by the bottom-layer cycle S_ϵ, which we refer to as the *big cycle*. The graph G' is the union of the big cycle with another set of edges, which we call **bucket edges**. The bucket edges are obtained by selectively including higher-level cycles as described in Section 2.2. An important property of the bucket edges is that the event that they expand a set is *independent* from the event that the cycle edges expand a set, so the effect of each class of edges can be analyzed separately. The idea of the proof is to show that all but a small number of node sets are expanded by the big cycle. The sets which are not expanded by the big cycle are expanded by the bucket edges with high probability.

2.1 The big cycle

We show that for most sets the edges of the big cycle S_ϵ alone suffice to show expansion. The remaining sets expand due to the bucket edges. For a set of vertices A denote by $\delta_0(A)$ the set of big cycle edges which have exactly one end point in A. The following Lemma is proven in [3]:

Lemma 2.2. *In a n-node skip graph, the number of sets A where $|A| = m < n/2$ and $|\delta_0(A)| \leq s$ is at most*

$$2\binom{m+1}{s}\binom{n-m-1}{s}$$

Take s to be ϵm for a sufficiently small ϵ. We have:

$$2\binom{m+1}{\epsilon m}\binom{n-m-1}{\epsilon m} \leq \exp\left(0.1 m \ln \frac{n}{m}\right) \leq \binom{n}{m}^{0.1} \qquad (1)$$

2.2 The bucket edges

The second subgraph is obtained by partitioning the nodes into a disjoint family of **buckets**, upper-level cycles S_σ that are not too small.[2] A cycle S_σ is a bucket if σ is a minimal prefix for which

[1]Throughout the paper the term 'with high probability' stands for probability at least $1 - n^{-c}$ for some $c > 0$ independent of n.

[2]Recall that S_σ is the doubly-linked list of all nodes whose membership vectors have σ as a prefix.

$|S_\sigma| \geq 10$ and either $|S_{\sigma 0}| < 10$ or $|S_{\sigma 1}| < 10$ (or both). Equivalently, the buckets are the cycles obtained by repeatedly splitting cycles, starting with the original cycle S_ϵ, by adding one bit at a time to the common prefix, stopping only when further divisions would yield cycles that are too small. The minimum bucket size is set to 10 for convenience in the proof, but other values may be chosen instead. In simulations, we show that a bucket size of 4 appears to be the best choice.

We call the edges that create the cycles of each bucket the **bucket edges** of the graph. The following observation motivates the division into buckets: Consider a given set A of nodes. Whenever there exists a bucket which contains a node in A and a node not in A, the bucket contributes at lease one edge to $\delta(A)$. Our aim therefore is to prove that with high probability there are at least $\epsilon|A|$ buckets that are partially covered by A; i.e. that A hits at least one element and misses at least one element from each bucket.

Lemma 2.3. *With high probability for all $1 \leq m \leq \frac{n}{2}$ the number of nodes in buckets which contain more than $100n^{\frac{1}{4}}m^{-\frac{1}{4}}$ nodes is at most $\frac{m}{10}$, where the probability is taken over the random choices of membership vectors.*

Before proving Lemma 2.3, we show how to deduce Theorem 2.1 given that the event described in the lemma occurs. Let ϵ be a small constant and assume the membership vectors are chosen such that the event in Lemma 2.3 occurs. For a set of nodes A denote by $\delta_1(A)$ the number of bucket edges with exactly one endpoint in A. Sets A for which $|\delta_1(A)| < \epsilon|A|$ are referred to as *bad sets*. In the following, for each $m \leq \frac{n}{2}$ we upper-bound the number of bad sets of cardinality m. We emphasize that we count the number of bad sets given a choice of membership vectors, i.e. with a predetermined allocation of nodes into buckets.

Call buckets which contain more than $100n^{\frac{1}{4}}m^{-\frac{1}{4}}$ nodes **large buckets**. The rest of the buckets are referred to as **small**. We do not take into account the expanding edges which belong to large buckets (thus over counting the number of bad sets). We count the number of bad sets of size m by first calculating the number of ways to choose nodes from the large buckets, then calculating the number of ways to choose nodes from buckets which are expanding (there are at most ϵm such buckets). Finally we calculate the number of ways we can choose nodes from non-expanding buckets. The total number of bad sets is upper bounded by the multiplication of these three estimates.

According to Lemma 2.3, there are at most $\frac{m}{10}$ nodes in large buckets, therefore there are at most $2^{m/10}$ possible ways in which the nodes in the large buckets are subsets of a bad set.

We now proceed with counting the number of combinations in which a bad set could be composed of nodes from the small buckets. Since each bucket contains at least 10 nodes, there at most $n/10$ buckets. We first choose the nodes which belong to expanding buckets. We count the number of ways to pick nodes from expanding buckets by first picking the buckets themselves. There are at most $\binom{n/10}{\epsilon m}$ ways to choose the ϵm small buckets that do expand. Now we count the number of ways to pick nodes from these buckets. Each small bucket is of size at most $100n^{\frac{1}{4}}m^{-\frac{1}{4}}$. It follows that there are at most $\binom{100\epsilon n^{\frac{1}{4}}m^{\frac{3}{4}}}{m}$ ways to choose vertices in these ϵm small buckets.

The remaining vertices are chosen from other buckets, with the restriction that each such bucket is either not hit by the set or is covered completely by it. Thus the number of ways to choose the remaining nodes is reduced to the number of ways to choose buckets. Each bucket is of size at least 10, therefore there are at most $\binom{n/10}{m/10}$ ways to choose vertices from the non expanding buckets. We conclude that the total number of bad sets is bounded by:

$$
\begin{aligned}
&2^{m/10}\binom{n/10}{\epsilon m}\binom{100\epsilon n^{1/4}m^{3/4}}{m}\binom{n/10}{m/10} \\
\leq\ & \exp\left(\begin{array}{l} (m/10)\ln 2 \\ +\epsilon m \ln\left(\frac{en}{10\epsilon m}\right) \\ +m\ln\left(100e\epsilon n^{1/4}m^{-1/4}\right) \\ +(m/10)\ln\left(\frac{en}{m}\right)\end{array}\right) \\
\leq\ & \exp\left(m\left(\begin{array}{l} \epsilon\ln\left(\frac{n}{m}\right)+\frac{1}{4}\ln\left(\frac{n}{m}\right)+\frac{1}{10}\ln\left(\frac{n}{m}\right) \\ +\frac{\ln 2}{10}+\epsilon\ln\left(\frac{e}{10\epsilon}\right)+\epsilon\ln\left(100e\epsilon\right)+\frac{1}{10}\end{array}\right)\right) \\
\leq\ & \exp\left(m\left(\epsilon\ln\left(\frac{n}{m}\right)+\frac{1}{4}\ln\left(\frac{n}{m}\right)+\frac{1}{10}\ln\left(\frac{n}{m}\right)+0.2\right)\right) \\
& \qquad [\text{for } \epsilon \leq 1/100] \\
\leq\ & \exp\left(0.8m\ln\frac{n}{m}\right). \qquad (2)
\end{aligned}
$$

where for the first inequality we use the fact that $\binom{n}{m} \leq \left(\frac{ne}{m}\right)^m$.

The total number of sets of size m is $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$, so only a small fraction of $\binom{n}{m}^{-0.2}$ of the sets are not expanded by the bucket edges. The key observation is that the large cycle depends only on the keys of the nodes and thus is *independent* from the division into buckets. In other words, all bad sets with respect to the large cycle are equally likely to be expanded by the bucket edges.

Inequality (1) states that the number of sets which are not expanded by the cycle edges is at most $\binom{n}{m}^{0.1}$. Inequality (2) implies that the probability a set of size m is not expanded by the bucket edges is at most $\binom{n}{m}^{-0.2}$. We conclude that the probability there exists a set of size m which is not expanding is at most $\binom{n}{m}^{-0.1}$. The proof of Theorem 2.1 is completed by union bounding these probabilities for all $2 \leq m \leq \frac{n}{2}$ and the error probability of Lemma 2.3.

We now proceed with the proof of Lemma 2.3:

Proof of Lemma 2.3. Let M denote the set of all prefixes of length $\lfloor \log n - \log\log n - 3 \rfloor$, so that $\frac{n}{16\log n} \leq |M| \leq \frac{n}{8\log n}$. For every $\sigma \in M$, let S_σ denote all the nodes that have σ as a prefix.

Lemma 2.4. *For every $\sigma \in M$, with high probability* $\log n \leq |S_\sigma| \leq 24\log n$.

Proof. This is a simple balls and bins argument. For each $\sigma \in M$, $|S_\sigma|$ has the Binomial distribution and $8\log n \leq \mathbb{E}[|S_\sigma|] \leq 16\log n$. By Chernoff's bound:

$$
\Pr[|S_\sigma| \leq \log n] \leq \exp\left(-\frac{\mathbb{E}[|S_\sigma|]}{4}\right) \leq \frac{1}{n^2}
$$
$$
\Pr[|S_\sigma| \geq 24\log n] \leq \exp\left(-\frac{\mathbb{E}[|S_\sigma|]}{4}\right) \leq \frac{1}{n^2}
$$

□

We conclude that w.h.p. all buckets are of size at most $24\log n$, therefore the lemma is correct if $m \leq \frac{n}{\log^4 n}$. Assume therefore that $m > \frac{n}{\log^4 n}$. As mentioned the buckets are established by repeatedly splitting cycles until further splits would create a small cycle. Suppose that during the procedure of splitting the cycles we have a cycle of size ℓ which corresponds to the prefix σ. The cycle does not split into two cycles (and thus becomes a bucket) if there are less than 10 nodes with prefix $\sigma.0$ or less than 10 nodes with prefix $\sigma.1$. Conclude that the probability a cycle of size ℓ is not partitioned into two cycles of size at least 10 is

$$2\left(\binom{\ell}{0}+\binom{\ell}{1}+\ldots+\binom{\ell}{10}\right)2^{-\ell}\leq 5\ell^{10}2^{-\ell}.$$

Let p_k denote the probability a node belongs to a bucket of size at least k. Once a node participates in more than $n-k$ partitions, it must belong to a bucket of size smaller than k, so we have:

$$p_k \leq \sum_{\ell=k}^{n} 5\ell^{10}2^{-\ell} \leq 20k^{10}2^{-k} \tag{3}$$

According to Lemma 2.4, we can divide the nodes into sets according to the first $\lfloor \log n - \log\log n - 3\rfloor$ bits of their prefix. Each set is of size at most $24\log n$, and there are at most $\frac{n}{8\log n}$ such sets. Denote by Z_i the random variable counting the number of nodes in the ith set which will eventually be in a bucket of size at least k. We know the following:

1. For each i it holds that $0 \leq Z_i \leq 24\log n$.
2. All the Z_i are mutually independent.
3. Inequality (3) implies that $\mathbb{E}[\sum Z_i] = np_k \leq 20nk^{10}2^{-k}$.

We use the following version of the Chernoff/Hoeffding bound:

Theorem 2.5. *For mutually independent random variables $Z_1,\ldots,Z_\ell$, where $Z_i\in[a,b]$*

$$\Pr\left[\left|\sum_{i=1}^{\ell} Z_i - \mathbb{E}\left[\sum_{i=1}^{\ell} Z_i\right]\right| \geq \ell\delta\right] \leq 2e^{-\frac{2\delta^2}{(b-a)^2}\cdot\ell}$$

Set $k = 100\left(\frac{n}{m}\right)^{\frac{1}{4}}$ and $\mu = \mathbb{E}[\sum Z_i] \leq 20nk^{10}2^{-k}$ and we have:

$$\Pr\left[\sum Z_i \geq \frac{m}{10}\right] \leq \Pr\left[\left|\sum Z_i - \mu\right| \geq \mu - \frac{m}{10}\right]$$

We can use Theorem 2.5 by setting $(b-a) = 24\log n$ and $\ell = \frac{n}{24\log n}$ and $\delta = (\frac{m}{10}-\mu)\frac{\log n}{n}$. We have:

$$\begin{aligned}
&\leq 2\exp\left(-\frac{1}{24^2\log^2 n}\cdot 2\left(\frac{m}{10}-\mu\right)^2\frac{\log^2 n}{n^2}\cdot\frac{n}{24\log n}\right)\\
&\leq 2\exp\left(-\left(\frac{m}{10}-\mu\right)^2\cdot\frac{1}{24^3 n\log n}\right) \qquad (4)
\end{aligned}$$

Next we bound μ as a fraction of m. Substitute $k = 100\left(\frac{n}{m}\right)^{\frac{1}{4}}$ and $\mu \leq 20nk^{10}2^{-k}$ and we have::

$$\begin{aligned}
\frac{m}{10}-\mu &\geq \frac{m}{10} - 2000n\left(\frac{n}{m}\right)^{\frac{10}{4}}\cdot 2^{-100\left(\frac{n}{m}\right)^{1/4}}\\
&\geq m\left(\frac{1}{10} - 2000\cdot 2^4\cdot 2^{-100}\right) \geq 0.09m
\end{aligned}$$

Now we complete the calculation of Equation (4) using the assumption that $m \geq \frac{n}{\log^4 n}$.

$$\leq 2 \exp\left(-(0.09m)^2 \cdot \frac{1}{24^3 n \log n}\right) \leq 2 \exp\left(-\frac{n}{20 \cdot 10^5 \ln^5 n}\right)$$

The proof of Lemma 2.3 is completed by union bounding for all m. □

2.3 Identifying the bucket edges

Each node can identify its bucket edges in $O(1)$ time in expectation and $O(\log n)$ time with high probability via the following procedure: initially each node sets its maximal edge as a bucket edge; i.e., assumes that the prefix of its bucket is the longest prefix for which it has an edge. Next each node performs a walk along the bucket's cycle to verify that the bucket's size is large enough i.e. that the bucket contains at least 10 nodes[3]. While performing this check, the node updates the other nodes along the cycle about the bucket edge. Now there are a few cases:

1. If the bucket is of size less than 10 then the prefix of the bucket is too long. So the node picks the next longest edge as its bucket edge and again counts the size of the bucket's cycle.
2. It may be that even though the bucket's size is more than 10 nodes, a node is informed that its current bucket edge is not valid and that it has to reduce the length of the bucket edge. This case occurs if the bucket which corresponds to the same prefix with the last bit flipped is too small; i.e. a different node performed case number (1).
3. If the bucket size is at least 10 and case (2) does not occur then the bucket edges are decided upon.

The running time of the algorithm is in the order of the size of the bucket, therefore the algorithm runs in expected constant time, and in logarithmic time with high probability.

3 How to sample a random node

Expanders are well known for having logarithmic mixing time (see, for example, [4, 12] for surveys of this area). The following theorem is standard (see for instance [1]):

Theorem 3.1. *Let $\hat{A} = \frac{1}{d}A$ be the transition matrix of a random walk on a d-regular graph with adjacency matrix A. Let α be the second-largest eigenvalue of A. Then for every initial distribution $\vec{p}$, it holds that*

1. $||\hat{A}^t\vec{p} - \vec{u}||_1 \leq \sqrt{n}\alpha^t \cdot ||\vec{u} - \vec{p}||_2$ where $|| \cdot ||_i$ stands for the L_i norm.

2. If $t > \frac{\log(1/\delta)}{\log(1/\alpha)}$ then for every node v it holds that $|(\hat{A}^t\vec{p})_v - \frac{1}{n}| \leq \delta$.

So we can obtain an approximately uniform sample from a skip graph (e.g. sampling each node with probability within $\delta = \frac{1}{2n}$ of uniform) by performing a random walk of length $\Theta(\log n)$ on the subgraph formed by cycle and bucket edges used in Theorem 2.1. In practice, the only limitation is that we need to be able to estimate the mixing time.

This depends both the second eigenvalue of the transition matrix and the value of $\log n$. Our simulations show (see Section 3.2) that the second eigenvalue is concentrated around 0.85. An estimate of $\log n$ can be easily derived through simple procedures, see for example [13, 17].

[3]If we allow buckets of size 2 then this part is not necessary

3.1 Speeding up the mixing time

Theorem 2.1 refers to a constant degree subgraph. One might hope that the logarithmic degree of the skip graph implies that using more edges would significantly decrease the mixing time. Unfortunately this is not the case.

Lemma 3.2. *With high probability there exists a set $A \subset V$ such that $\frac{n}{2} - \log n\sqrt{n} \le |A| \le \frac{n}{2}$, and $|\delta(A)| \le |A|$.*

Proof. For $\tau \in \{0, 1\}$ Let A_τ be the set of vertices that have τ as the first bit of their membership vector. Assume w.l.o.g that $|A_0| \le |A_1|$. We have that w.h.p. $|A_0| \ge \frac{n}{2} - \log\sqrt{n}$ and that $|\delta(A_0)| \le |A_0|$. □

Lemma 3.2 implies that the edge expansion of the entire skip graph is $O(1)$, so for every subgraph of the skip graph, the mixing time is bounded by $\Omega(\log n)$. Furthermore since the conductance of the set A_0 is $O(\frac{1}{\log n})$ the mixing time of the entire skip graph is $\Omega(\ln n \ln\ln n)$. It may be the case however that adding a small set of edges to the subgraph would improve the mixing time by a constant.

We speed up the sampling using a different approach. In a skip graph it is possible to lookup a node with a specific *membership vector*. This property is used to hot-start a random walk and thus reduce the complexity of sampling. The procedure is as follows:

1. Choose a vector m of $3\log n$ random bits. Look up a node[4] with the longest prefix which agrees with m. Call that node v.

2. Perform a random walk according on the expanding subgraph, starting at v.

Let $\vec{p}$ be the distribution formed by the first phase of the algorithm. Our goal is to bound $||\hat{A}^t\vec{p} - u||_1$ when t is the length of the random walk. Let $\vec{\epsilon} = \vec{u} - \vec{p}$. Theorem 3.1 states that $||\hat{A}^t\vec{p} - u||_1 \le \sqrt{n}\alpha^t||\vec{\epsilon}||_2$. We calculate a bound on $||\vec{\epsilon}||_2$. First note that with high probability it holds that $p_v \le \frac{2\log n}{n}$ for every node v. This is true since w.h.p. every vector of length $\log(n/2\log n)$ is the prefix of at least one node. We conclude that:

$$||\vec{\epsilon}||_2 \le \sqrt{n \cdot \left(\frac{2\log n}{n}\right)^2} \le \frac{2\log n}{\sqrt{n}}$$

and that $||\hat{A}^t\vec{p} - u||_1 \le 2\alpha^t \log n$ so in order for the distance to be at most δ we need $t \ge \frac{\log(\frac{2\log n}{\delta})}{\log(1/\alpha)}$. For example, if we take $\delta = 1/100$ and $\alpha = 0.85$ we get that $t \ge 33 + 4.3\log\log n$. As before, it is important to note that these are only upper bounds, and it may be that the walk mixes much faster. Indeed, in Section 3.2 we show that once a random prefix is reached, a very short walk yields an almost uniform sample.

The complexity of Step (1) depends upon the algorithm used for searching a prefix. It takes $O(\log n)$ steps w.h.p. to find a prefix when the node is searched greedily (this is basically identical to the algorithm for finding the correct neighbors upon joining). The constant hiding behind the $O-$notation is about 1. The number of hops could be further reduced by using the *Neighbor of Neighbor* algorithm of Manku *et al* [14].

[4]Ties may be broken arbitrarily.

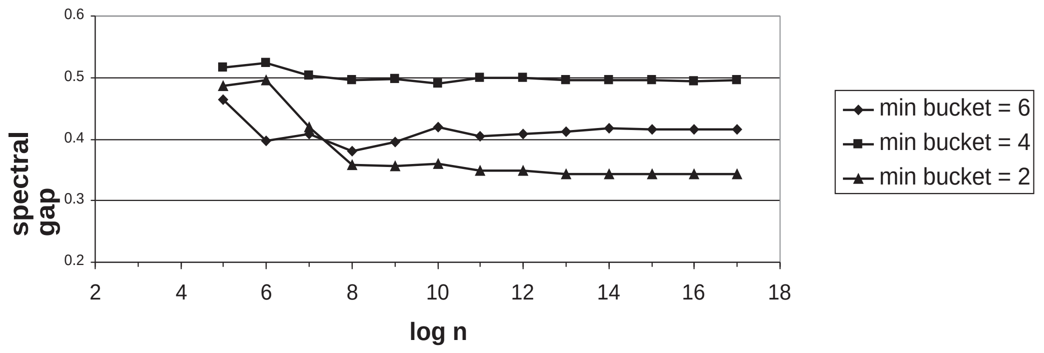

Figure 1: The spectral gap of the Laplacian as a function of log n, using various values for the minimum bucket size.

3.2 Empirical evidence

In the simulations we constructed the expanding subgraph. We calculated the second eigenvalue of the Laplacian (which is the gap between the first and second eigenvalue of the adjacency matrix). In Figure 1 we sketch the spectral gap as a function of $\log n$. Each entry is the average of 5 simulations. The first observation is that the value of the second eigenvalue is extremely concentrated; when $n = 256$ the difference between the smallest gap measured and the largest was under 0.1 for all different bucket sizes. When $n = 2^{16}$ the difference between the smallest and largest measurement was less than 0.03. Simulations show that the largest gap is achieved when the minimum bucket size is 4. In this case the spectral gap is roughly 0.495. The value of α from Theorem 3.1 could be calculated as: $\frac{4-0.495}{4} \approx 0.87$. The second eigenvalue of a Ramanujan graph, that is, a graph with the largest possible spectral gap is bounded by $2\sqrt{3} \approx 3.46$. In other words the largest spectral gap we could have hoped for is $4 - 3.46 = 0.54$. In this case the value of α is $\frac{3.46}{4} = 0.866$, so the value of α we achieve is about 0.004 from optimal.

Aspnes and Shah [3] give experimental results for the effects of random node failures of various probabilities p for $n = 2^{17}$. Their results show that most surviving nodes of a skip graph continue to be fully connected for p up to approximately 0.6, with rapidly increasing fragmentation for larger values of p. This is consistent with the large spectral gap we observed. For small values of p, the high expansion holds the graph together, while for larger values the probability that an individual node becomes isolated after losing all of its $O(\log n)$ neighbors rises rapidly.

Next we checked the quality of the mixing by directly calculating the distribution over the nodes when starting from some arbitrary vertex. The vector $\hat{A}^t\vec{p}$ was explicitly calculated when $\vec{p}$ puts weight 1 on the first vertex. The simulations have $n = 2^{18}$ and bucket size of at least 4. Table 1 summarizes the results. The *minimum weight* column indicates the probability weight of the node least likely to be sampled, as a fraction of $\frac{1}{n}$. The *maximum weight* is the analog for the heaviest vertex. When the walk is of length $2.5 \log n$ the variation distance from uniform is only 0.018 and all vertices are sampled with probability at least $0.9\frac{1}{n}$. When the walk is of length $3.5 \log n$ all vertices are sampled with probability at most $1.33\frac{1}{n}$.

It is important to note that even though our choice of expanding subgraph achieves an almost optimal spectral bound, in practice it may not necessarily be the optimal choice as far as mixing is considered. For instance it may be that a random walk which uses several bucket sizes together

Walk Length	*minimum weight*	*maximum weight*	*variation distance*
$1 \log n$	0.1	990	0.49
$1.5 \log n$	0.42	162	0.185
$2 \log n$	0.74	35	0.059
$2.5 \log n$	0.9	7.7	0.018
$3 \log n$	0.96	3.21	0.0055
$3.5 \log n$	0.99	1.33	0.0015
$4 \log n$	0.996	1.12	0.0005

Table 1: Quality of mixing when $n = 2^{18}$ and buckets are of size at least 4.

Walk Length	*minimum weight*	*maximum weight*	*variation distance*
$2 \approx 0.1 \log n$	0.05	4.95	0.14
$4 \approx 0.2 \log n$	0.41	2.75	0.08
$7 \approx 0.4 \log n$	0.67	1.72	0.04
$11 \approx 0.6 \log n$	0.83	1.29	0.017
$14 \approx 0.8 \log n$	0.9	1.16	0.01
$18 = \log n$	0.95	1.08	0.005

Table 2: Quality of hot-started mixing when $n = 2^{18}$, buckets are of size at least 4 and the walk starts from a random prefix.

would mix faster. When designing such heuristics one must bear in mind that the mixing time is *not* monotone in the number of edges: indeed, the entire skip graph mixes slower than the expanding subgraph.

Random walk with a hot start: We simulated a random walk starting from a *random prefix.* Table 2 summarizes the results. It could be seen that once a random prefix is reached, a random walk of length $7 = \lfloor 0.4 \log n \rfloor$ samples each node with probability at most $1.71\frac{1}{n}$ and at least $0.67\frac{1}{n}$. Thus, this is by far the fastest known sampling algorithm.

Estimating the expansion The simulations above provide a lower bound on the expansion of the graph. The following theorem is proven in [2]:

Theorem 3.3. *If λ is the second largest eigenvalue of a $d-$regular graph G with n vertices, then the node expansion of G is at least $\frac{2(d-\lambda)}{3d-2\lambda}$.*

Plugging in $d = 4$ and $\lambda = 4 - 0.495 = 3.505$ we get that the node expansion of the expanding subgraph is at least 0.18. The bound in Theorem 3.3 is probably not tight in our case. Furthermore, since the expansion is monotone in the number of edges, we expect the expansion ratio of the entire skip graph to be larger.

3.3 The maximal edge heuristic

Another possible subgraph to consider is the one composed of the big cycle edges and *maximal edges*. An edge (u, v) is said to be *maximal* if for either u or v it corresponds to the longest prefix that yields a nonempty cycle. Given Theorem 2.1, it is natural to conjecture that a random walk on these edges would mix rapidly. The main advantage of this scheme is that the maximum edge of each node is immediately identifiable without any overhead. A disadvantage is that the degrees of nodes in the resulting subgraph are not uniform, which produces a nonuniform stationary distribution.

However, this nonuniform distribution can be corrected by applying rejection sampling to the more frequently sampled high-degree nodes. We give empirical evidence that this heuristic converges quickly, achieving a distribution within 1% of uniform in just $5 \log n$ hops.

Figure 2 plots the degree distribution of three different graph sizes. Each plot is the average of five simulations. The results of the simulations were very well concentrated. About 97.5% of the nodes have degrees 3 or 4. Degree 5 nodes are about 2.5% of the nodes. The remaining degrees could be found in less than 0.1% of the nodes, and in no simulation have we encountered a node with degree larger than 7. The average degree in all 15 simulations was between 3.275 and 3.285. Nodes with higher degree have a higher probability of being sampled by the walk. As mentioned previously, this could be fixed by using rejection sampling based on the node degrees: when a random walk ends at a node of degree d, it samples the node with probability $\frac{3}{d}$ (which is 1 most of the time) and initiates a new random walk with the remaining probability. The expected length of the walk increases by a factor of approximately 16/15.

The mixing time is estimated by calculating the second eigenvalue of the Laplacian. Figure 3 charts the spectral gap of the Laplacian as a function of the graph size. Each value is the average of 15 simulations. It could be seen that a spectral gap exists, and stands at about 0.256. This implies an $O(\log n)$ mixing time using standard results.

To check the constant on the mixing time, we ran simulations and checked the variation distance between the random walk sample and the stationary distribution. The results are summarized is Table 3. A walk of length $5 \log n$ yields a variation distance of less than $\frac{1}{100}$ from stationary. As discussed above, we can make the stationary distribution uniform by rejecting sampling at the cost of an additional $\frac{16}{15}$ factor on the time.

Walk Length	*variation distance*
$1 \log n$	0.72
$2 \log n$	0.29
$3 \log n$	0.095
$4 \log n$	0.025
$5 \log n$	0.007

Table 3: The variation distance between the random walk distribution and the stationary distribution, when $n = 2^{18}$, and the maximum edge is used.

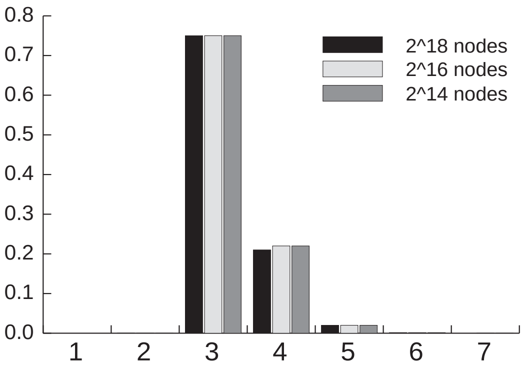

Figure 2: The distribution of the degrees when the maximum edge is taken.

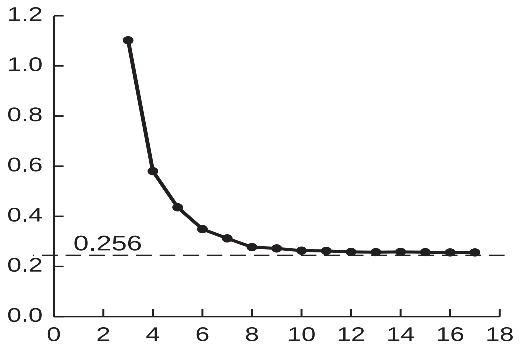

Figure 3: The spectral gap of the Laplacian as a function of log n, using the maximum edge heuristic.

4 Open problems

A natural question that arises is whether the techniques used to prove expansion for skip graphs generalize to other finger-table-based data structures like Chord [18].

It is not hard to see that the maximal edge heuristic fails to produce an expanding subgraph in a Chord ring. For example, consider the set consisting of all nodes with IDs in the intervals $[0, 0.25)$ and $[0.50, 0.75)$. The longest edges from each component, which connect a node with ID x to the node whose ID is nearest to $x + 0.5 \bmod 1$, reach only nodes in or adjacent to the other component. Similar constructions show that choosing lower (but still fixed) layers of the Chord finger tables also give poor expansion.

However, this does not rule out the possibility of a more sophisticated argument might demonstrate high expansion. The actual expansion ratio of a Chord ring thus remains open.

References

[1] Miklos Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing (STOC)*, pages 132–140, 1987.

[2] Noga Alon and Vitali Milman. λ_1, isoperimetric inequalities for graphs and superconcentrators. *Journal of Combinatorial Theory*, 38:73–88, 1985.

[3] James Aspnes and Gauri Shah. Skip graphs. In *fourteenth ACM SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[4] Fan R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, 92:1–212, 1997.

[5] Joel Friedman. A proof of Alon's second eigenvalue conjecture. In *Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing (STOC)*, pages 720–724, 2003.

[6] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, 2004.

[7] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of USITS, USENIX.*, 2003.

[8] Valerie King, Scott Lewis, and Jared Saia. On algorithms for choosing a random peer. Unpublished manuscript, 2005.

[9] Valerie King and Jared Saia. Choosing a random peer. In *Proceedings of the 23nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 125–130, 2004.

[10] Ching Law and Kai-Yeung Siu. Distributed construction of random expander graphs. In *Proceedings of IEEE INFOCOM*, 2003.

[11] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M. Hellerstein. The case for a hybrid P2P search infrastructure. In *3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 141–150, 2004.

[12] L. Lovasz. *Random Walks on Graphs: A Survey*, pages 1–46. Kesthely, 1993.

[13] Gurmeet S. Manku. Routing networks for distributed hash tables. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 133–142, 2003.

[14] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 54–63, 2004.

[15] G. A. Margulis. Explicit constructions of concentrators. *Problemy Peredachi Informatsii*, (9(4)), October - December 1973.

[16] Moni Naor and Udi Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 50–59, 2003.

[17] Moni Naor and Udi Wieder. A simple fault tolerant distributed hash table. In *Second International Workshop on Peer-to-Peer Systems*, pages 88–97, February 2003.

[18] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[19] Kevin C. Zatloukal and Nicholas J. A. Harvey. Family trees: an ordered dictionary with optimal congestion, locality, degree, and search time. In *fifteenth ACM SIAM Symposium on Discrete Algorithms (SODA)*, pages 308–317, 2004.