# The Expansion and Mixing Time of Skip Graphs with Applications

James Aspnes* Udi Wieder†

February 7, 2005

**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 could be used to sample a node in the skip graph in a highly efficient manner. We also show that the expansion property could be used to load balance the skip graph quickly. Finally it is shown that the skip graph could serve as an unstructured P2P system, thus it is a good candidate for a hybrid P2P system.

## Regular presentation

*Yale University Department of Computer Science. Email: aspnes@cs.yale.edu. Supported in part by NSF grants CCR-0098078, CNS-0305258, and CNS-0435201.

†The Weizmann Institute of Science, Department of Applied Math and Computer Science. Rehovot, Israel. Email: udi.wieder@weizmann.ac.il.

# 1 Introduction

Skip graphs [6] or SkipNets [10] 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 [17]. Skip graphs have an advantage over DHTs in the sense 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 [6] 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 (which are analyzed in Sections 3 and 4) 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 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 unstructured P2P systems which are expanders permit more efficient searches than simple flooding [9]. 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, and the literature is abundant with variations of expanding networks. In the context of dynamic P2P networks we are aware of only two previous constructions. Naor and Wieder [19] build an overlay network that emulates the Margulis [18] 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 [19] 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 [14]. They suggest building $d$ random Hamiltonian cycles, which have an optimal spectral gap w.h.p. and are thus expanding [7]. 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 construction does

not support a lookup operation.[1] Their construction also needs a sampling protocol as a primitive, and assumes that samples are obtained by performing a random walk. 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.

**Sampling schemes** In the context of P2P systems, a random walk sampling scheme was previously suggested by Law and Siu [14] (as mentioned above) and by Gkantsidis [9]. Obtaining good samples using such random walks requires *a priori* knowledge of the spectral gap of the graph. In Section 3.2, we show 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.

A different sampling scheme for DHTs was suggested by King and Saia [13]. Their scheme yields an *exact* uniform distribution and runs in expected logarithmic time. Recent work by King, Lewis and Saia [12] shows 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 $2 \log n$ 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 $1, 2, \ldots, n$; as 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 $S_\epsilon$ so that node $i$ is connected to $i - 1 \mod n$ and $i + 1 \mod n$. For each finite bit-vector $\sigma$, an additional circular doubly-linked list $S_\sigma$ is constructed by taking all nodes whose membership vectors have $\sigma$ as a prefix, and linking adjacent nodes in the lexicographic key order. More formally, let $m(x) \restriction k$ 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 $m(x) \restriction k = m(y) \restriction k$, and either (a) $m(z) \restriction k \neq m(x) \restriction k$ for each $z$ between $x$ and $y$, or (b) $m(z) \restriction k \neq m(x) \restriction k$ 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 $S_\sigma$.

## 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 or edges with exactly one endpoint in $U$.

[1] In order to support lookup their construction would need logarithmic degree and then a lookup takes $\Theta(\log^2 n)$ hops.

**Theorem 2.1.** *With high probability*[2] *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 $n$ ).*

**Remark:** We do not state what is the largest $\epsilon$ for which Theorem 2.1 is correct. Indeed, in the proof we are liberal in making $\epsilon$ as small as necessary. A lower bound on 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.

The subgraph $G'$ consists of two types of edges. In a skip graph, all the nodes are connected by the bottom-layer cycle $S_\epsilon$. The graph $G'$ is the union of these cycle edges and another set of edges, which we call **bucket edges**, that 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 the probability a set $U \subset V$ does not expand by the cycle edges *and* by the bucket edges is sufficiently small.

## 2.1 The cycle edges

We show that for most sets the cycle edges 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 cycle edges which have exactly one end point in $A$. The following Lemma is proven in [6]:

**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 $\epsilon m$ for a sufficiently small $\epsilon$. we have:

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

## 2.2 The bucket edges

The bucket edges are obtained by partitioning the nodes among a disjoint family of **buckets**, upper-level cycles $S_\sigma$ that are not too small.[3] A cycle $S_\sigma$ is a bucket if $\sigma$ is a minimal prefix for which $|S_\sigma| \geq 10$ 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 $S_\epsilon$, 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, other values may be chosen as well. 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 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.

[2] Throughout the paper the term 'with high probability' stands for probability $1 - n^{-\delta}$ for some $\delta > 0$

[3] Recall that $S_\sigma$ is the doubly-linked list of all nodes whose membership vectors have $\sigma$ as a prefix.

**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 $\frac{m}{10}$.*

Before proving Lemma 2.3, we show how to deduce Theorem 2.1 given that the event described in the lemma occurs. Let $\epsilon$ be a small constant. We calculate for how many sets of size $m$ the buckets do not contribute $\epsilon m$ edges to the expansion.

Call buckets which contain more than $100n^{\frac{1}{4}}m^{-\frac{1}{4}}$ nodes **large buckets**. The rest of the buckets will be referred to as **small**. We do not count edges caused by large buckets (thus overcounting the number of bad sets). According to Lemma 2.3, there are at most $\frac{m}{10}$ nodes in large buckets, therefore there are at most $2^{m/10}$ ways to place the nodes in the large buckets. Since each bucket contains at least 10 nodes, there at most $n/10$ buckets. There are at most $\binom{n/10}{\epsilon m}$ ways to choose the $\epsilon m$ small buckets that do expand. Each small bucket is of size at most $100n^{\frac{1}{4}}m^{-\frac{1}{4}}$. It follows that there are at most $\binom{\epsilon 100 n^{\frac{1}{4}} m^{\frac{3}{4}}}{m}$ ways to place the vertices in these $\epsilon m$ small buckets. The remaining vertices are scattered in other buckets, with the restriction that each such bucket is either not hit by the set or is covered completely by it. Each bucket is of size at least 10, therefore there are at most $\binom{n/10}{m/10}$ ways to choose the location of the rest of the vertices. We conclude that the total number of bad sets is bounded by:

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

In 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 $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$. 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 states that the probability a set of size $m$ is not expanded by the bucket edges is at most $\binom{n}{m}^{-0.2}$. Conclude that the probability there exists a set of size $m$ which is not expanding is at most $\binom{n}{m}^{-0.1}$. Now the theorem is proven 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 all the 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 $B_\sigma$ denote all the nodes that have $\sigma$ as a prefix.

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

*Proof.* This is a simple balls and bins argument. For each $\sigma \in M$, $|B_\sigma|$ has the Binomial distribution and $8\log n \leq \mathbb{E}[|B_\sigma|] \leq 16\log n$. Use Chernoff's bound to complete the proof. □

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 to the contrary that $m > \frac{n}{\log^4 n}$. Suppose further that during the procedure of creating the

buckets we have a bucket of size $\ell$ which corresponds to the prefix $\sigma$. The bucket does not split into two buckets 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 bucket of size $\ell$ is not partitioned into two buckets 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 an element 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 $i$th 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:

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

Next we bound $\mu$ 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[4]. 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 converted 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

A random walk on a regular expander mixes rapidly, and may be used to sample a node in the skip graph efficiently and simply. In the following it is convenient to think of distributions over the nodes as vectors. We say that $\vec{p}$ is a **distribution vector** if $p_i \geq 0$ for all $0 \leq i < n$, and $\sum p_i = 1$. Let $\vec{u}$ denote the probability vector of the uniform distribution over the nodes; i.e. $\vec{u} = (\frac{1}{n}, \ldots, \frac{1}{n})$. Let $\vec{p}$ be some arbitrary distribution over the nodes. A useful measure for the distance between two distribution is the **variation distance**[5] which is defined to be half of the $L_1$ distance between their vectors; i.e. $\Delta(\vec{p}, \vec{u}) = \frac{1}{2}||\vec{p} - \vec{u}||_1 = \frac{1}{2}\sum_v |p_v - \frac{1}{n}|$. Assume a random walk is performed on a $d-$regular graph, starting from some arbitrary initial distribution $\vec{p}$ (distribution $\vec{p}$ may of course put all its weight on a single vertex). Let $A$ be the adjacency matrix of the $d-$regular graph and let $\hat{A} = \frac{1}{d}A$. The largest eigenvalue of $A$ is $d$ (and $\vec{u}$ is an eigenvector), so the largest eigenvalue of $\hat{A}$ is $1$. Denote by $\alpha$ the second largest eigenvalue of $\hat{A}$. Now for every integer $t$ the vector $\hat{A}^t\vec{p}$ is the distribution over the nodes after performing $t$ steps of a random walk. It holds (see [2]) that if the graph has a node or edge expansion which is bounded away from $0$ then $\alpha$ will be bounded away from one.[6] The following theorem is well known (see for instance [1]):

**Theorem 3.1.** *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 $|(\hat{A}^t\vec{p})_v - \frac{1}{n}| \leq \delta$. In particular, if $\delta = \frac{1}{2n}$ then the probability each node is sampled by the walk is in the range $[\frac{1}{2n}, \frac{3}{2n}]$.*

Theorem 3.1 combined with Theorem 2.1 implies that a long enough random walk along the subgraph formed by the cycle edges and the bucket edges yields an approximately uniform sample. The length of the random walk may depend upon the application. If it is required that each node be sampled with probability

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

[5] There are many ways to define distance between distributions. Variation distance is the most useful in our context.

[6] We slightly abuse notation. The statement is meaningful only when discussing families of graphs with $n \to \infty$ and not a single graph, which is of course our case.

which is within factor 2 of uniform then, by the second assertion of the theorem, a walk of length $\frac{1+\log n}{\log(1/\alpha)}$ suffices. For a variation distance between the sample and the uniform distribution bounded by $\epsilon$, according the the first assertion a walk of length $t = \frac{\log n + 2\log(1/\epsilon)}{2\log(1/\alpha)}$ is enough (note that if $\vec{p}$ puts all its weight on one node then $||\vec{p} - \vec{u}||_2 \approx 1$).

The running time to obtain a fixed variation distance depends upon the second eigenvalue and upon $\log n$. Simulations show (see Section 3.2) that the second eigenvalue is concentrated around $0.85$. An estimation of $\log n$ could be easily derived through simple procedures, see for example [20, 16].

## 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 $\frac{n}{2} - \log n\sqrt{n} \leq |A| \leq \frac{n}{2}$, and $|\delta(A)| \leq |A|$.*

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

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.

### 3.1.1 Sampling in $O(\log n/\log\log n)$ time:

The unique properties of the skip graph, in particular its support for the lookup operation, can be used to hot-start a random walk, and thus reduce the complexity of sampling. Let $\vec{u}$ denote the uniform probability. In the following we show a procedure that for a given $\delta > 0$ samples a node with distribution $\vec{p}$ such that $||u - p||_1 \leq \delta$. The expected running time is $O(\log n/\log\log n)$ where the constant depends upon $\delta$. The procedure is as follows:

1. Choose a vector $m$ of $3\log n$ random bits. look up the node[7] with the longest prefix which agrees with $m$. Call that node $v$.
2. Perform a random walk according to the bucket scheme, starting at $v$ and of length $O(\log n/\log\log n)$.

Let $\vec{p}$ be the distribution formed by the first phase of the algorithm. We need to bound $||\hat{A}^t\vec{p} - u||_1$ when $t = O(\log n/\log\log n)$. Let $\vec{\epsilon} = \vec{u} - \vec{p}$. Theorem 3.1 states that $||\hat{A}^t\vec{p} - u||_1 \leq \sqrt{n}\alpha^t||\vec{\epsilon}||_2$. We calculate a bound on $||\vec{\epsilon}||_2$. First note that with high probability it holds that $p_v \leq \frac{2\log n}{n}$ 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:

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

and that $||\hat{A}^t\vec{p} - u||_1 \leq 2\alpha^t \log n$ so in order for the distance to be at most $\delta$ we need $t \geq \frac{\log\left(\frac{2\log n}{\delta}\right)}{\log(1/\alpha)}$. For example, if we take $\delta = 1/100$ and $\alpha = 0.85$ we get that $t \geq 33 + 4.3\log\log n$. As before, it is important to

[7] In case of several such nodes, any one of them suffices.

note that these are upper-bounds only. 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 edges when joining). We show it takes expected $O(\log n/\log\log n)$ steps to look up a prefix when the *Neighbor of Neighbor* algorithm of Manku, Naor and Wieder [17] is used. Assume that each node is aware of the prefixes of its neighbor's neighbors (i.e. $O(\log^2 n)$ nodes). Each step of the algorithm (referred to as a NoN step) the message is passed to the neighbor's neighbor with the longest prefix which matches the target (ties broken arbitrarily); i.e. each NoN step is composed of 2 routing steps. The proof of the following is given in the full paper.

**Theorem 3.3.** *For every binary vector $\sigma$, the expected time it takes the NoN algorithm to find a node with the longest prefix which agrees with $\sigma$ is $O\left(\frac{\log n}{\log\log n}\right)$.*

### 3.2 Empirical evidence

We ran simulations in which the bucket edges and cycle edges were constructed. 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 $5$ simulation only. The first observation is that the value of the second eigenvalue is extremely concentrated (which is why the average of $5$ simulations is enough). When $n = 256$ the difference between the smallest gap measured and the largest was under $0.1$ for all different bucket sizes. When $n = 65536$ 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 $\alpha$ from Theorem 3.1 could be calculated as: $\frac{4-0.495}{4} \approx 0.87$.

Next we checked the quality of the mixing by 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 $3.5\log n$ then all vertices are sampled with probability at most $1.33\frac{1}{n}$.

It is important to note that the bucket edges are not necessarily the optimal choice. For instance it may be that a random walk which uses several bucket sizes together would mix faster. When designing such heuristics one must bare 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 may be used to give lower bound on the expansion of the graph. The following theorem is proven in [3]:

**Theorem 3.4.** *If $\lambda$ 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 3.4 is probably not tight in our case, furthermore, since the expansion is monotone in the number of edges, we expect the expansion of the skip graph to be a larger constant.

### 3.3 The maximal edge heuristic

Another possible subgraph to consider is the one composed of the 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 by without any overhead. A disadvantage is that the degrees of nodes in the resulting subgraph are not uniform, which produces a nonuniform stationary distribution. In the full paper, we show that this nonuniform distribution can be corrected by applying rejection sampling to the more frequently sampled high-degree nodes. We also give empirical evidence (some of which is summarize in Figures 2 and 3 and Table 3) that this heuristic converges quickly, achieving a distribution within $1\%$ of uniform in just $5 \lg n$ steps.

## 4 Applications

Consequences of expansion in skip graphs can be divided between those that use the expansions directly, like fault tolerance, and those that depend on the resulting rapid mixing of random walks. We discuss both below.

### 4.1 Fault tolerance

Aspnes and Shah [6] showed via simulations that skip graphs are highly resilient to random failure of nodes. The expansion property of skip graphs gives the theoretical support to these empirical findings.

When adversarial faults are considered we have the following conclusion: If $k$ nodes are deleted in an *adversarial* manner, then the largest connected component would have $\Omega(n - O(k))$ nodes. In other words, even if an adversary deletes a constant fraction of nodes, still a constant fraction of the nodes would remain connected.

### 4.2 Hitting large sets

As seen, a random walk of length $O(\log n)$ in an expander yields a random point. One of the most interesting and appealing properties of expander graphs is that in some sense a random walk of length $o(\log n)$ yields $\Theta(\log n)$ random samples. Clearly the nodes hit by the random walk are highly correlated, yet for the purpose of hitting a set of nodes, they behave as if they were independent random samples. The following theorem was proven by Ajtai, Komlós and Szemerédi [1]; it can also be found in Alon and Spencer's book [4].

**Theorem 4.1 (Ajtai, Komlós and Szemerédi).** *Let $G = (V, E)$ be a $d-$regular graph on $n$ vertices, and suppose that each of its eigenvalues but the first one is at most $\lambda$. Let $C$ be a set of $cn$ vertices of $G$. Then, for every $\ell$, the number of walks of length $\ell$ in $G$ that avoid $C$ does not exceed $(1 - c)n((1 - c)d + c\lambda)^\ell$.*

There are $nd^\ell$ walks of length $\ell$, which means that if we pick a random starting point for the walk, then the probability of avoiding the set $C$ is at most:

$$\frac{(1-c)n((1-c)d + c\lambda)^\ell}{nd^\ell} = (1-c) \cdot \left(\frac{(1-c)d + c\lambda}{d}\right)^\ell \leq \left(\frac{(1-c)d + c\lambda}{d}\right)^\ell$$

We know that a random walk of length $O(\log n)$ reaches a random point, therefore for any initial point, a random walk of length $\ell + O(\log n)$ avoids $C$ with probability at most $\left(\frac{(1-c)d+c\lambda}{d}\right)^{\ell} = \left(1-(1-\frac{\lambda}{d})c\right)^{\ell}$. Note that the probability the set $C$ is avoided by $\log n$ independent samples is $(1-c)^{\log n}$. The theorem states that this probability can be achieved by a walk of length $O(\log n)$.

We use the random-walk property in a somewhat different setting than the one described by Aspnes and Shah in [5]. We begin by briefly sketching this setting and then continue with applications of random walks.

### 4.2.1 The skip graph as a peer-to-peer data storage system

So far, we have assumed that each node of the graph represents a data item to be searched for. This allows for a simple implementation of *range queries* over the data set. Unfortunately, it also comes with a price: each data item is put separately in the system and requires roughly $O(\log n)$ communication links. Distributed Hash Tables which do not support range queries group elements together such that the total number of communication links is a function of the number of servers rather than data items (c.f. [22, 21, 23, 19]). A natural way to group data items is to let each server hold a contiguous *segment* of the key space. Each server puts only one element of that segment in the skip graph (this data item effectively serves as the key of the skip graph node), and holds the rest in some internal data structure (which allows range queries). Now each node corresponds to a server in the system and not to a data item, and the capability to range query the data set is preserved.

### 4.2.2 Load balancing

The problem with the construction above is that it might be the case that different servers hold different fraction sizes of the data, thus dividing the load unevenly. A large fraction of the data set inflicts load not only due to the memory needed to store the items, but also because a server which holds many data items would be queried more often. It is therefore desirable that nodes share the data items as evenly as possible. Aspnes *et al* [5] and Karger and Ruhl [11] suggest various algorithms to deal with this problem. The algorithms they design apply a re-balancing mechanism that involves heavily loaded nodes actively seeking lightly loaded nodes. We take a different approach and suggest a simple *Join* algorithm, which preserves load balancing as long as the distribution from which data items receive their keys does not change often.

The algorithm we suggest is very simple. A server that wishes to join the skip graph performs a random walk of length $\Omega(\log n)$, recording the number of data items held by each node it encounters. It then picks a segment in the key space such that it splits the load of the most heavily loaded of these nodes. Consider a set $H$ of heavily loaded nodes. According to Theorem 4.1, the probability $H$ is hit by a walk of length $\Omega(\log n)$ is equal to the probability it would be hit by $\Omega(\log n)$ *independent* random samples.

We simulated the second, simpler scenario. In our simulation we put $2^{27}$ data items in a single node, and then added $2^{18}$ nodes one by one. The $k$th node to enter sampled $\ln k$ nodes uniformly and independently and split the load of the heaviest sample. The resulting division of data items between the nodes was strikingly balanced. While the average load is $2^9$, no node had a load larger than $2^{12}$. Clearly this simulation oversimplifies the model: in particular, it does not take into account deletion of nodes and the dynamics of the data items themselves. Yet if we assume that the distribution of data item names is fixed, then it is reasonable to assume that the random walk would be a good load balancing heuristic. Testing the random walk algorithm in more realistic scenarios is an important future goal.

#### 4.2.3 Locating highly replicated data items

Assume that some data item is immensely popular and appears at a large fraction of the nodes. Theorem 4.1 implies that a random walk would hit a node holding the popular item fast. It is shown in [15] that for popular data items an exhaustive search (as in Gnutella) is more efficient than an exact search using a DHT. It is shown in [9] that when the underlying graph is an expander, a random walk is more efficient than exhaustive search. We conclude that a skip graph may serve as an excellent *hybrid* data structure; i.e., may serve as a structured and unstructured P2P system simultaneously.

#### 4.2.4 Gathering statistics

Assume we want to estimate what fraction of the skip graph nodes run Linux, or more generally we want to estimate the fraction of nodes which have some property. A natural approach would be to sample $\Theta(\log n)$ nodes randomly and use the sample to estimate the fraction. Typically a Chernoff bound is then used to show that the answer is approximately correct w.h.p. David Gillman [8] proved a theorem in the spirit of Theorem 4.1 and showed that one random walk of length $O(\log n)$ may serve to produce $\Theta(\log n)$ random samples. In the following we let $\lambda$ denote the spectral gap of the normalized adjacency matrix (i.e. $1-\alpha$), let $C$ be a set of $cn$ nodes.

**Theorem 4.2.** *Let $t_k$ be the number of visits to $C$ in $k$ steps of a random walk, starting from some arbitrary distribution, then* $\Pr[|t_k - ck| \geq \gamma] \leq ne^{-\frac{\gamma^2\lambda}{20k}}$.

Note that $ck$ is the expected number of visits in the set. It follows that if $c$ is a constant and $k$ is $O(\log n)$ then w.h.p. the estimation of $c$ could be arbitrarily close to $c$ itself.

## 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. Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986.

[3] Noga Alon and Vitali Milman. $\lambda_1$, isoperimetric inequalities for graphs and superconcentrators. *Journal of Combinatorial Theory*, 38:73–88, 1985.

[4] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 2000.

[5] James Aspnes, Jonathan Kirsch, and Arvind Krishnamurthy. Load balancing and locality in range-queriable data. In *Twenty-Third ACM Symposium on Principles of Distributed Computing (PODC)*, pages 115–124, 2004.

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

[7] 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.

[8] David Gillman. A chernoff bound for random walks on expander graphs. *Siam Journal on Computing*, 27(4):1203–1220, 1998.

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

[10] 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.

[11] David Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2004.

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

[13] 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.

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

[15] 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)*, 2004.

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

[17] 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.

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

[19] 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.

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

[21] Anthony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[22] 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.

[23] Ben Y. Zhao and John Kubiatowicz. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB CSD 01-1141, University of California at Berkeley, Computer Science Department, 2001.

# Figures

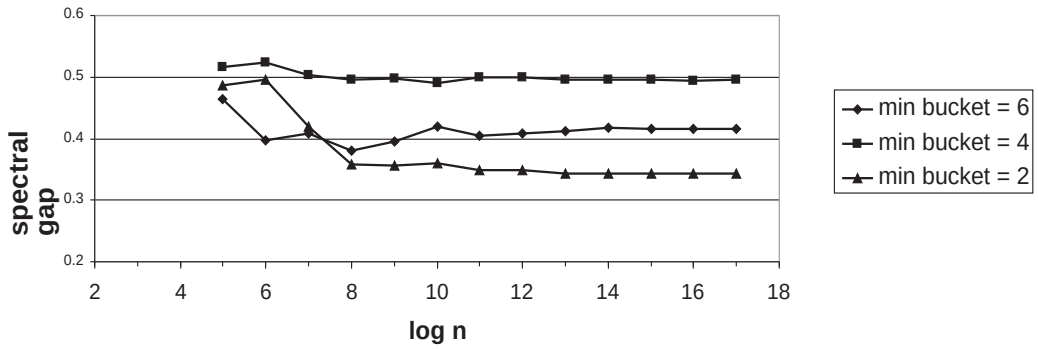


Figure 1: The spectral gap of the Laplacian as a function of log $n$, using various values for the minimum bucket size.

| *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.

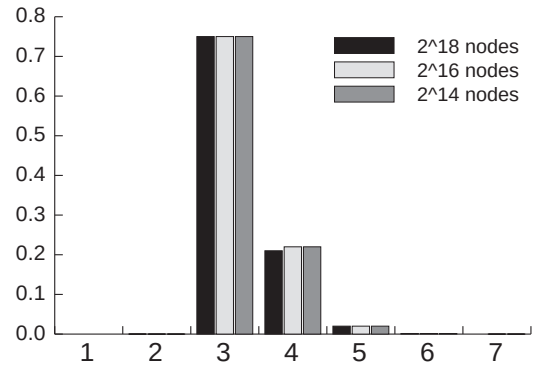


Figure 2: The distribution of the degrees when the maximum edge is taken.

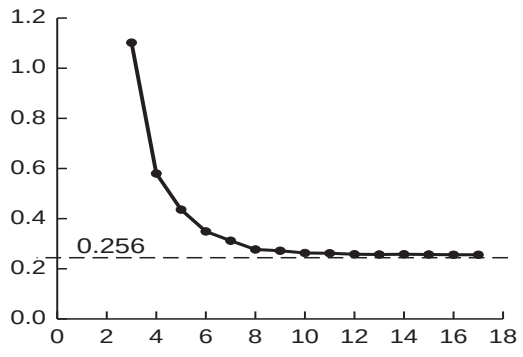


Figure 3: The spectral gap of the Laplacian as a function of $\log n$, using the maximum edge heuristic.

| *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.