Fault-tolerant Routing in Peer-to-peer Systems*

James Aspnes[†] Zoë Diamadi[‡] Gauri Shah[§]

July 26, 2002

Abstract

We consider the problem of designing an overlay network and routing mechanism that permits finding resources efficiently in a peer-to-peer system. We argue that many existing approaches to this problem can be modeled as the construction of a random graph embedded in a metric space whose points represent resource identifiers, where the probability of a connection between two nodes depends only on the distance between them in the metric space. We study the performance of a peer-to-peer system where nodes are embedded at grid points in a simple metric space: a one-dimensional real line. We prove upper and lower bounds on the message complexity of locating particular resources in such a system, under a variety of assumptions about failures of either nodes or the connections between them. Our lower bounds in particular show that the use of inverse power-law distributions in routing, as suggested by Kleinberg [6], is close to optimal. We also give heuristics to efficiently maintain a network supporting efficient routing as nodes enter and leave the system. Finally, we give some experimental results that suggest promising directions for future work.

1 Introduction

Peer-to-peer systems are distributed systems without any central authority and with varying computational power at each machine. We study the problem of locating resources in such a large network of heterogeneous machines that are subject to crash failures. We describe how to construct distributed data structures that have certain desirable properties and allow efficient resource location.

Decentralization is arguably a critical feature of such a system as any central server not only provides a vulnerable point of failure but also wastes the power of the clients. Equally important is scalability: the cost borne by each node must not depend too much on the network size and should ideally be proportional, within polylogarithmic factors, to the amount of data the node seeks or provides. Since we expect nodes to arrive and depart at a high rate, the system should be resilient to both link and node failures. Furthermore, disruptions to parts of the data structure should self-heal to provide self-stabilization.

*This is an extended version of the paper appearing in the proceedings of the *Twenty-First ACM Symposium on Principles of Distributed Computing*, 2002

[†]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `aspnes@cs.yale.edu`. Supported by NSF grants CCR-9820888 and CCR-0098078.

[‡]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `diamadi@cs.yale.edu`. Supported in part by ONR grant N00014-01-1-0795.

[§]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `gauri.shah@yale.edu`. Supported by NSF grants CCR-9820888 and CCR-0098078.

Our approach provides a hash table-like functionality, based on keys that uniquely identify the system resources. To accomplish this, we map resources to points in a metric space either directly from their keys or from the keys' hash values. This mapping dictates an assignment of nodes to metric-space points. We construct and maintain a random graph linking these points and use greedy routing to traverse its edges to find data items. The principle we rely on is that failures leave behind yet another (smaller) random graph, ensuring that the system is robust even in the face of considerable damage. Another compelling advantage of random graphs is that they eliminate the need for global coordination. Thus, we get a fully-distributed, egalitarian, scalable system with no bottlenecks.

We measure performance in terms of the number of messages sent by the system for a search or insert operation. The self-repair mechanism may generate additional traffic, but we expect to amortize these costs over the search and insert operations. Given the growing storage capacity of machines, we are less concerned with minimizing the storage at each node but the space requirements are small. The information stored at a node consists only of a network address for the location of each neighbor.

The rest of the paper is organized as follows. Section 2 explains our abstract model in detail, and Section 3 describes some of the existing peer-to-peer systems. We prove our results for routing in Section 4, followed by a discussion on how to construct the random graph in Section 5. We present experimental results in Section 6 and conclusions and future work in Section 7.

2 Our approach

The idea underlying our approach consists of three basic parts: (1) embed resources as points in a metric space, (2) construct an appropriate random graph by linking these points, and maintain it dynamically as the network and the resources provided by it change, and (3) efficiently locate individual data items by traversing the edges of the graph according to a greedy routing strategy. More specifically, let R be a set of resources spread over a large, heterogeneous network N. For each resource $r \in R$, $owner(r)$ denotes the node in N that provides r and $key(r)$ denotes the resource's key. Let K be the set of all possible keys. We map each resource onto a point v in a metric space (V, d), where V is the point set and d is the distance metric. We assume a hash function $h : K \to V$, and let resource r map to the point $v = h(key(r))$, as shown in Figure 1. We assume that h generates random-looking locations for the resources that the keys identify. Note that via this resource embedding a node n is mapped onto the set $V_n = \{v \in V : \exists r \in R, v = h(key(r)) \wedge (owner(r) = n)\}$, namely the set of metric-space points assigned to the resources the node provides.

Our next step is to carefully construct a directed random graph, embedded in V, which the nodes in N will use to locate resources in the network. We would like the outgoing edges of each vertex to be chosen independently of those of other vertices. This will allow for the graph to be constructed locally: each node $n \in N$ generates the outgoing edges for all vertices in V_n. An edge $(v, u) \in V_n \times V_m$ simply denotes that n knows that m is the network node that provides the resource mapped to u; hence, we can view the graph as a virtual overlay network of information, pieces of which are stored locally at each node. The next obvious question is how this information is located. We assume that each newly-arrived node n is initially connected to some other node in N. For each $v \in V_n$, n chooses the vertices that will be the sinks of v's outgoing edges. It then constructs each edge by executing the search algorithm to locate the resource that is mapped to that sink of that edge. If the metric space is not populated densely enough, the choice of a sink may result in a vertex corresponding to an absent resource. In that case, n chooses the closest neighbor to the original sink that corresponds to a present resource. Moving to nearby vertices will introduce some

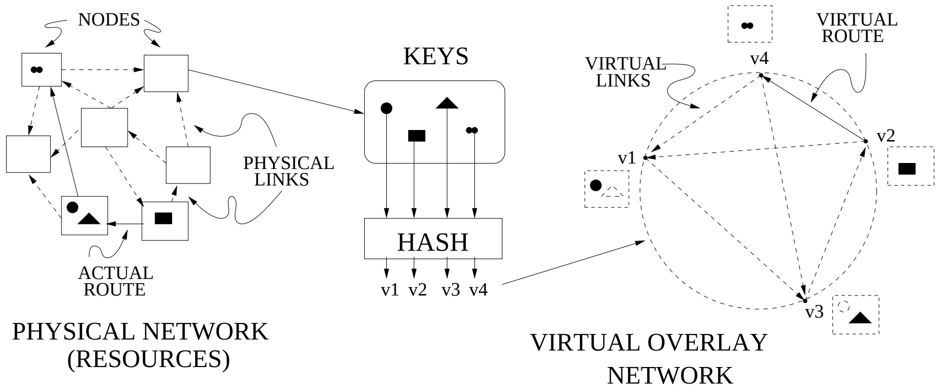


Figure 1: An example of the metric-space embedding.

bias in the edge distribution, but the error does not appear to be large. A more detailed description of the graph construction is given in Section 5.

Having constructed the overlay network of information, we can now use it for resource location. As new nodes arrive, old nodes depart, and existing ones alter the set of resources they provide or even crash, the resources available in the distributed database change. At any time t, let $R^t \subseteq R$ be the set of available resources and I^t be the corresponding overlay network constructed by means of the aforementioned procedure. A request by node n to locate resource r at time t is served in a simple, localized manner: n calculates the metric-space point v that corresponds to r, and a request message is then routed over I^t from the vertex in V_n (that is closest to v) to v itself.[1] Each node needs only local information, namely its set of neighbors in I^t, to participate in the resource location. The routing strategy used is a greedy one, that is, the request message is always forwarded to the node mapped to a metric-space point as close to v as possible. The problem of resource location is thus translated into routing on random graphs embedded in a metric space.

To a first approximation, our approach is similar to the "small-world" routing work by Kleinberg [6], in which points in a two-dimensional grid are linked by edges drawn from a normalized power-law distribution (with exponent 2), and routing is done by having each node route a packet to its neighbor closest to the packet's destination. Kleinberg's approach is somewhat brittle because it assumes a constant number of edges leaving each node. Getting good performance depends both on having a complete two-dimensional grid of nodes and on very carefully adjusting the exponent of the random edge distribution. We are not as interested in keeping the degree down and accept a larger degree to get more robustness. We also cannot assume a complete grid, since fault-tolerance is one of our main objectives, and since nodes are mapped to points in the metric space based on what resources they provide, there may be missing nodes.

The use of random graphs is partly motivated by a desire to keep the data structure scalable and the routing algorithm as decentralized as possible, as random graphs can be constructed locally without global coordination. Another important reason is that random graphs are by nature robust against node failures: a node-induced subgraph of a random graph is generally still a random graph; therefore, the disappearance of a vertex, along with all its incident edges (due to failure of one of the machines implementing the data structure) will still allow routing while the repair mechanism is trying to heal the damage. The repair mechanism also benefits from the use of random graphs, since most random structures require less work to maintain their much weaker invariants compared

[1]Note that since R^t generally changes with time, and may specifically change while the request is being served, the request message may be routed over a series of different overlay networks I^{t_1}, I^{t_2}, $\ldots$, I^{t_k}.

to more organized data structures.

Embedding the graph in a metric space has the very important property that the only information needed to locate a resource is the location of its corresponding metric-space point. That location is permanent, both in the sense of being unaffected by disruption of the data structure, and easily computable by any node that seeks the resource. So, while the pattern of links between nodes may be damaged or destroyed by failure of nodes or of the underlying communication network, the metric space forms an invulnerable foundation over which to build the ephemeral parts of the data structure.

Finally, we should note our model of adversarial behavior. We have already alluded to possible crash failures of the network nodes. We also consider link failures. The hash function is assumed to populate the metric space evenly; the adversary is an oblivious one.

3 Current peer-to-peer systems

There are several peer-to-peer systems in development right now which appear to be vastly different from each other from a distant perspective but in fact, are inherently very similar in spirit. Some of the first-generation systems such as Napster [9], Gnutella [2] and Freenet [1] are good examples of existing decentralized Internet systems that have inspired the development of more sophisticated second-generation systems such as CAN [12], Chord [13] and Tapestry [3].

Most of the peer-to-peer systems in widespread use are clearly not scalable. Napster has a central server that services requests for shared resources even though the actual resource transfer takes place between the peer requesting the resource and the peer providing it, without involving the central authority. However, this has several disadvantages including a vulnerable single point of failure, wasted computational power of the clients as well as unscalability. Gnutella actually decentralizes the initial resource location as well by flooding the network with the resource request. This is not only unscalable and noisy, but it also does not guarantee finding a resource, even though it exists in the network, if it just happens to be *out of reach* of the requesting node. While the use of super-peers [8] ameliorates the problem somewhat in practice, it does not improve performance in the limit. Some of the later systems overcome most of these drawbacks and provide more efficient decentralized resource location.

Tapestry [3] uses Plaxton's algorithm [11] as the core for routing which is based on suffix-based hypercube routing. Each node has a unique n-bit identifier divided into l levels of $w = n/l$ bits each. A node x will have l sets of 2^w neighbors each, such that each set i, for $0 \leq i < l$, will have nodes with identifiers as follows: i common digits with x's identifier, followed by all possible 2^w values for the $(i+1)$-th digit and any of the 2^w possible values for the remaining digits. Routing is then done deterministically by *resolving* the identifier one digit at a time moving towards the target node. Thus each node stores $O(\lg n)$ state and delivery time is also $O(\lg n)$.

CAN [12] partitions a d-dimensional coordinate space into *zones* that are owned by nodes. Keys are mapped to points in the space and then stored at the node which owns the zone in which the point is located. Each node maintains an $O(d)$ state about its neighbors which abut with the node only in 1 dimension. Greedy lookup by forwarding messages to the neighbor closest to the target zone takes $O(dn^{1/d})$ time. Additional techniques are used for data replication, improved information retrieval as well as addition and deletion of nodes.

Chord [13] provides a hash-table functionality by mapping nodes to identities of m bits placed around a *modulo* 2^m *identifier circle*. Each nodes stores a routing table with m entries such that the i-th entry stores the key of the first node succeeding it by at least 2^{i-1} on the identifier circle. Each resource is also mapped onto the identifier circle and stored at the first node succeeding the location that it maps to. Routing is done greedily to the farthest possible node in the routing table and it is not hard to see that this gives an $O(\lg n)$ delivery time with n nodes in the system.

The recurrent underlying theme beneath all these systems is that there is some variant of an overlaying metric space in which the nodes are embedded. The location of a resource in this metric space is decided by its key. Each node maintains some information about its neighbors in the metric space. The routing is then simply done by forwarding packages to neighbors closer to the target node with respect to a suitable metric distance defined on the space. In CAN, the metric space is explicitly defined as the coordinate space which is covered by the zones and the distance metric used is simply the Euclidean distance. On the other hand for Chord, the nodes can be thought of being embedded on grid points on a real circle. Distances measured along the circumference of the circle provide the required distance metric. In Tapestry, we can think of the nodes being embedded on a real line and the identifiers are simply the locations of the nodes on the real line. Euclidean distance is used as the metric distance for greedy forwarding to nodes with identifiers closest to the target node. This inherent common structure leads to similar results for the performance of such networks. In this paper, we explain why most of these systems achieve similar performance guarantees by describing a general setting for such overlay metric spaces, although most of our results are obtained from one-dimensional spaces.

The fault-tolerant properties of these systems are not very well-defined. Each system provides a repair mechanism in the event of failures but makes no performance guarantees till this mechanism kicks in. Indeed for a system of such a large size, where nodes appear and vanish at very fast rates, the resilience of the system to repeated and concurrent failures is a desirable and important property. Our experiments show that with our overlay space and linking strategies, the system performs reasonably well even with a large number of failures.

4 Routing

In this section, we present our lower and upper bounds on routing.

4.1 Tools

Some of our upper bounds will be proved using a well-known upper bound of Karp *et al.*[4] on probabilistic recurrence relations. We will restate this bound as Lemma 1, and then show how a similar technique can be used to get *lower bounds* with some additional conditions in Theorem 2.

Lemma 1 ([4]) *The time $T(X_0)$ needed for a nonincreasing real-valued Markov chain $X_0, X_1, X_2, X_3 \ldots$ to drop to 1 is bounded by*

$$T(X_0) \leq \int_1^{X_0} \frac{1}{\mu_z} dz, \tag{1}$$

when $\mu_z = \mathrm{E}[X_t - X_{t+1} : X_t = z]$ is a nondecreasing function of z.

This bound has a nice physical interpretation. If it takes one second to jump down μ_x meters from x, then we are traveling at a rate of μ_x meters per second during that interval. When we zip past some position z, we are traveling at the average speed μ_x determined by our starting point $x \geq z$ for the interval. Since μ is nondecreasing, using μ_z as our estimated speed underestimates our actual speed when passing z. The integral computes the time to get all the way to zero if we use μ_z as our instantaneous speed when passing position z. Since our estimate of our speed is low (on average), our estimate of our time will be high, giving an upper bound on the actual expected time.

We would like to get lower bounds on such processes in addition to upper bounds, and we will not necessarily be able to guarantee that μ_z as defined in Lemma 1, will be a nondecreasing function of z. But we will still use the same basic intuition: The average speed at which we pass z is at most the maximum average speed of any jump that takes us past z. We can find this maximum speed by taking the maximum over all $x > z$; unfortunately, this may give us too large an estimate. Instead, we choose a threshold U for "short" jumps, compute the maximum speed of short jumps of at most U for all x between z and $z + U$, and handle the (hopefully rare) long jumps of more than U by conditioning against them. Subject to this conditioning, we can define an upper bound m_z on the average speed passing z, and use essentially the same integral as in (1) to get a lower bound on the time. Some additional tinkering to account for the effect of the conditioning then gives us our real lower bound, which appears in Theorem 2 below, as Inequality (8).

Theorem 2 *Let $X_0, X_1, X_2, \ldots$ be Markov process with state space S, where X_0 is a constant. Let f be a non-negative real-valued function on S such that, for all t,*

$$\Pr[f(X_t) - f(X_{t+1}) \geq 0 : X_t] = 1. \tag{2}$$

Let U and ϵ be constants such that for any $x > 0$,

$$\Pr[f(X_t) - f(X_{t+1}) \geq U : X_t = x] \leq \epsilon. \tag{3}$$

Let

$$\tau = \min\{t : f(X_t) = 0\}. \tag{4}$$

For each x with $f(x) > 0$, let $\mu_x > 0$ satisfy

$$\mu_x \geq \mathrm{E}[f(X_t) - f(X_{t+1}) : X_t = x, f(X_t) - f(X_{t+1}) < U]. \tag{5}$$

Now define

$$m_z = \sup\{\mu_x : x \in S, f(x) \in [z, z + U)\} \tag{6}$$

and define

$$T(x) = \int_0^{f(x)} \frac{1}{m_z} dz \tag{7}$$

Then

$$\mathrm{E}[\tau] \geq \frac{T(X_0)}{\epsilon T(X_0) + (1 - \epsilon)} \tag{8}$$

Proof: Define

$$Y_t = \begin{cases} T(X_t) & \text{if } f(X_{t'}) - f(X_{t'+1}) < U \text{ for all } t' < t, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The idea is that Y_t jumps to zero immediately if a long jump occurs. We will show that even with such overeager jumping, Y_t does not drop too quickly. The intuition is that the chance of a long jump reduces Y_t by at most an expected $\epsilon Y_t \leq \epsilon Y_0$, while the effect of short jumps can be bounded by applying the definition of T.

Let $\mathcal{F}_t$ be the σ-algebra generated by $X_0, X_1, \ldots X_t$. Let A_t be the event that $f(X_t) - f(X_{t+1}) < U$; i.e. that the jump from $f(X_t)$ to $f(X_{t+1})$ is a short jump. Now compute

$$\begin{aligned} E[Y_t - Y_{t+1} : \mathcal{F}_t] &= \Pr[\overline{A_t} : \mathcal{F}_t](Y_t - 0) + (1 - \Pr[\overline{A_t} : \mathcal{F}_t]) \operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t] \\ &\leq \Pr[\overline{A_t} : \mathcal{F}_t] Y_0 + (\epsilon - \Pr[\overline{A_t} : \mathcal{F}_t]) Y_0 + (1 - \epsilon) \operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t] \\ &= \epsilon Y_0 + (1 - \epsilon) \operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t] \end{aligned} \tag{10}$$

Now let us bound $\operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t]$. Expanding the definitions (7) and (9) gives

$$\operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t] = \operatorname{E}\left[\int_{f(X_{t+1})}^{f(X_t)} \frac{1}{m_z} dz : \mathcal{F}_t, A_t\right] \tag{11}$$

Now, conditioning on A_t means that $f_(X_{t+1}) > f(X_t) - U$ and thus $z > f(X_t) - U$ for the entire range of the integral. It follows that $f(X_t)$ lies in the half-open interval $[z, z+U)$ for each such z, from which we have $m_z \geq \mu_{f(X_t)}$ from (6). Inverting gives $\frac{1}{m_z} \leq \frac{1}{\mu_{f(X_t)}}$, and plugging this inequality into (11) gives

$$\begin{aligned} \operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, A_t] &\leq \operatorname{E}\left[\int_{f(X_{t+1})}^{f(X_t)} \frac{1}{\mu_{f(X_t)}} dz : \mathcal{F}_t, A_t\right] \\ &= \frac{1}{\mu_{f(X_t)}} \operatorname{E}[f(X_t) - f(X_{t+1}) : \mathcal{F}_t, A_t] \\ &\leq \frac{1}{\mu_{f(X_t)}} \mu_{f(X_t)} \\ &= 1. \end{aligned} \tag{12}$$

Applying (12) to (10) gives

$$\operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t] \leq \epsilon Y_0 + (1 - \epsilon). \tag{13}$$

We have now shown that Y_t drops slowly on average. To turn this into a lower bound on the time at which it first reaches zero, define $Z_t = Y_t + \min(t, \tau)\,(\epsilon Y_0 + (1 - \epsilon))$. Conditioning on $t < \tau$, observe that

$$\begin{aligned} \operatorname{E}[Z_t - Z_{t+1} : \mathcal{F}_t, t < \tau] &= \operatorname{E}[Y_t - Y_{t+1} : \mathcal{F}_t, t < \tau] - (\epsilon Y_0 + (1 - \epsilon)) \\ &\leq (\epsilon Y_0 + (1 - \epsilon)) - (\epsilon Y_0 + (1 - \epsilon)) \\ &= 0. \end{aligned}$$

Alternatively, if $t \geq \tau$ we have

$$\operatorname{E}[Z_t - Z_{t+1} : \mathcal{F}_t, t \geq \tau] = 0.$$

In either case, $\operatorname{E}[Z_t - Z_{t+1} : \mathcal{F}_t] \leq 0$, implying $Z_t \leq \operatorname{E}[Z_{t+1} : \mathcal{F}_t]$. In other words, $\{Z_t, \mathcal{F}_t\}$ is a submartingale.

Because $\{Z_t, \mathcal{F}_t\}$ is a submartingale, and τ is a stopping time relative to $\{\mathcal{F}_t\}$, we have $Z_0 = Y_0 \leq \operatorname{E}[Z_\tau] = \operatorname{E}[0 + \tau\,(\epsilon Y_0 + (1 - \epsilon))] = (\epsilon Y_0 + (1 - \epsilon)) \operatorname{E}[\tau]$. Solving for $\operatorname{E}[\tau]$ then gives

$$\operatorname{E}[\tau] \geq \frac{Y_0}{\epsilon Y_0 + (1 - \epsilon)} = \frac{T(X_0)}{\epsilon T(X_0) + (1 - \epsilon)}.$$

∎

4.2 Lower bound on greedy routing

We will now show a lower bound on the expected time taken by greedy routing on a random graph embedded in a line, where each node in the graph has expected outdegree at most ℓ, and the probability that a node at position x is connected to positions $x - \Delta_1, x - \Delta_2, \ldots, x - \Delta_k$ depends only on the set $\Delta = \{\Delta_1, \ldots, \Delta_k\}$ and not on x and is independent of the choice of outgoing edges for other nodes.[2]

We consider to two variants of the greedy routing algorithm. Without loss of generality, we assume that the target of the search is labeled 0. In *one-sided greedy routing*, the algorithm never traverses a link that would take it past its target. So if the algorithm is currently at x and is trying to reach 0, it will move to the node $x - \Delta_i$ with the smallest non-negative label. In *two-sided greedy routing*, the algorithm chooses a link that minimizes the distance to the target, without regard to which side of the target the other end of the link is. In the two-sided case the algorithm will move to a node $x - \Delta_i$ whose label has the smallest absolute value, with ties broken arbitrarily. One-sided greedy routing can be thought of as modeling algorithms on a graph with a boundary when the target lies on the boundary, or algorithms where all links point in only one direction (as in Chord).

Our results are stronger for the one-sided case than for the two-sided case. With one-sided greedy routing, we show a lower bound of $\Omega(\ln n/(\ell \ln \ln n))$ on the time to reach 0 from a point chosen uniformly from the range 1 to n that applies to any link distribution. For two-sided routing, we show a lower bound of $\Omega(\ln n/(\ell^2 \ln \ln n))$, with some constraints on the distribution. We conjecture that these constraints are unnecessary, and that $\Omega(\ln n/(\ell \ln \ln n))$ is the correct lower bound for both models.

In general, we assume that each node is connected to its immediate neighbors; the simplest way to model this is to require that ± 1 appear in Δ.

We will begin by developing machinery that will be useful in the proofs of both the one-sided and two-sided lower bounds.

4.2.1 Link sets: notation and distributions

First we describe some notation for Δ sets. Write each Δ as

$$\{\Delta_{-s}, \ldots \Delta_{-2}, \Delta_{-1} = -1, \Delta_1 = 1, \Delta_2, \ldots \Delta_t\},$$

where $\Delta_i < \Delta_j$ whenever $i < j$. Each Δ is a random variable drawn from some distribution on finite sets; the individual Δ_i are thus in general *not* independent. Let Δ^- consist of the s negative elements of Δ and Δ^+ consist of the t positive elements. Formally define $\Delta_{-i} = -\infty$ when $i > s$ and $\Delta_i = +\infty$ when $i > t$.

For one-sided routing, we make no assumptions about the distribution of Δ except that $|\Delta|$ must have finite expectation and Δ always contains 1. For two-sided routing, we assume that Δ is generated by including each possible δ in Δ with probability p_δ, where p is symmetric about the origin (i.e., $p_\delta = p_{-\delta}$ for all δ), $p_1 = p_{-1} = 1$, and p is unimodal, i.e. nonincreasing for positive δ and nondecreasing for negative δ.[3] We also require that the events $[\delta \in \Delta]$ and $[\delta' \in \Delta]$ are pairwise independent for distinct δ, δ'.

[2]We assume that nodes are labeled by integers, and identify each node with its label to avoid excessive notation.

[3]These constraints imply that $p_0 = 1$; formally, we imagine that 0 is present in each Δ but is ignored by the routing algorithm.

4.2.2 The aggregate chain S^t

For a fixed distribution on Δ, the trajectory of a single initial point X^0 is a Markov chain $X^0, X^1, X^2, \ldots$, with $X^{t+1} = s(X^t, \Delta^t)$, where Δ^t determines the outgoing links from the node reached at time t and s is a *successor function* that selects the next node $X^{t+1} = X^t - \Delta_i^t$ according to the routing algorithm. Note that the chain is Markov, because the presence of ± 1 links guarantees that no node ever appears twice in the sequence, and so each new node corresponds to a new choice of links.

From the X^t chain we can derive an *aggregate chain* that describes the collective behavior of all nodes in some range. Each state of the aggregate chain is a contiguous sets of nodes whose labels all have the same sign; we define the sign of the state to be the common sign of all of its elements. For one-sided routing each state is either $\{0\}$ or an interval of the form $\{1 \ldots k\}$ for some k. For two-sided routing the states are more general The aggregate states are characterized formally in Lemma 4.

Given a contiguous set of nodes S and a set Δ, define

$$S_{\Delta i} = \{x \in S : s(x, \Delta) = x - \Delta_i\}.$$

The intuition is that $S_{\Delta i}$ consists of all those nodes for which the algorithm will choose Δ_i as the outgoing link. Note that $S_{\Delta i}$ will always be a contiguous range because of the greediness of the algorithm. Now define, for each $\sigma \in \{-, 0, +\}$:

$$S_{\Delta i \sigma} = \{x \in S_{\Delta i} : \operatorname{sgn} s(x, \Delta) = \sigma\}.$$

Here we have simply split $S_{\Delta i}$ into those nodes with negative, zero, or positive successors.

For any set A and integer δ write $A - \delta$ for $\{x - \delta : x \in A\}$.

We will now build our aggregate chain by letting the successors of a range S be the ranges $S_{\Delta i \sigma} - \Delta_i$ for all possible Δ, i, and σ. As a special case, we define $S^{t+1} = \{0\}$ when $S^t = \{0\}$; once we arrive at the target, we do not leave it. For all other S^t, we let

$$\Pr\left[S^{t+1} = S^t_{\Delta i \sigma} - \Delta_i : \Delta\right] = \frac{|S^t_{\Delta i \sigma}|}{|S^t|}, \tag{14}$$

and define the unconditional transition probabilities by averaging over all Δ.

Lemma 3 shows that moving to the aggregate chain does not misrepresent the underlying single-point chain:

Lemma 3 *Let X^0 be drawn uniformly from the range S^0. Let Y^t be a uniformly chosen element of S^t. Then for all x and t,* $\Pr[X^t = x] = \Pr[Y^t = x]$.

Proof: Clearly the lemma holds for $t = 0$. Fix S^{t-1}, and consider two methods for generating Y^t. The first generates Y^t directly from Y^{t-1} and shows that Y^t generated in this way has the same distribution as X^t. The second generates Y^t from S^t as describe in the lemma and produces the same distribution on Y^t as the first.

In the first method, we choose Y^{t-1} uniformly from S^{t-1}, choose a random Δ^{t-1}, and compute $s(Y^{t-1}, \Delta^{t-1}$. Here the transition rule applied to Y^{t-1} is the same as for X^{t-1}, so under the induction hypothesis that Y^{t-1} and X^{t-1} are equal in distribution, so are Y^t and X^t.

In the second method, we again choose a random Δ^{t-1} and then choose S^t by choosing some $S^{t-1}_{\Delta i \sigma}$ in proportion to its size, let $S^t = S^{t-1}_{\Delta i \sigma} - \Delta_i$, and then let Y^t be a uniformly chosen element of S^t. We can implement the choice of $S^{t-1}_{\Delta i \sigma}$ by choosing some Y^{t-1} uniformly from S^{t-1} and picking

$S^{t-1}_{\Delta i\sigma}$ as the subrange that contains Y^{t-1}; and we can simplify the task of choosing Y^t by setting it equal to $Y^{t-1} - \Delta_i$, since conditioning on $Y^{t-1} \in S^{t-1}_{\Delta i\sigma}$ leaves Y^{t-1} with a uniform distribution. But by implementing the second method in this way, we have reduced it to the first, and the lemma is proved. ∎

Lemma 4 justifies our earlier characterization of the aggregate state spaces:

Lemma 4 *Let $S^0 = \{1 \dots n\}$ for some n. Then with one-sided routing, every S^t is either $\{0\}$ or of the form $\{1 \dots k\}$ for some k; and with two-sided routing, every S^t is an interval of integers in which every element has the same sign.*

Proof: By induction on t. For one-sided routing, observe that $S^{t-1}_{\Delta i-}$ is always empty, as the routing algorithm is not allowed to jump to negative nodes. If $S^t = S^{t-1}_{\Delta i0} - \Delta_i$, then $S^t = \{\Delta_i\} - \Delta_i = \{0\}$. Otherwise $S^t = S^{t-1}_{\Delta i+} - \Delta_i$; but since $S^{t-1} = \{1 \dots k\}$ for some k, if it contains any point x greater than Δ_i it must contain $\Delta_i + 1$; thus $\min(S^{t-1}_{\Delta i+} = \Delta_i + 1$ and so $\min(S^t)$ becomes 1.

The result for the two-sided case is immediate from the fact that $S^t = S^{t-1}_{\Delta i\sigma} - \Delta_i$ combined with the definition of $S^{t-1}_{\Delta i\sigma}$. ∎

The advantage of the aggregate chain over the single-point chain is that, while we cannot do much to bound the progress of a single point with an arbitrary distribution on Δ, we can show that the size of S^t does not drop too quickly given a bound ℓ on $\mathrm{E}[|\Delta|]$. The intuition is that each successor set of size $a^{-1}|S^t|$ or less occurs with probability at most a^{-1}, and there are at most 3ℓ such sets on average.

Lemma 5 *Let $\mathrm{E}[|\Delta|] \leq \ell$. Then for any $a \geq 1$, in either the one-sided or two-sided model,*

$$\Pr\left[|S^{t+1}| \leq a^{-1}|S^t|\right] \leq 3\ell a^{-1}. \tag{15}$$

Proof:
First note that if $a^{-1}|S^t| < 1$, then $\Pr\left[|S^{t+1}| \leq a^{-1}|S^t|\right] = 0$. So we can assume that $a^{-1}|S^t| \geq 1$ and in particular that $a \leq |S^t|$.

Conditioning on Δ, there are at most $3|\Delta|$ non-empty sets $S^t_{\Delta i\sigma}$. If $|S^t_{\Delta i\sigma}| \leq a^{-1}|S^t|$, then $|S^t_{\Delta i\sigma}|$ is chosen with probability at most a^{-1} by (14). Thus the probability of choosing any of the at most $3|\Delta|$ sets $S^t_{\Delta i\sigma}$ of size at most $a^{-1}|S^t|$ is at most $3|\Delta|a^{-1}$.

Now observe that

$$\begin{aligned}
\Pr\left[|S^{t+1}| \leq a^{-1}|S^t|\right] &\leq \sum_d \Pr\left[|\Delta| = d\right] 3da^{-1} \\
&= 3a^{-1}\,\mathrm{E}\left[|\Delta|\right] \\
&\leq 3\ell a^{-1}.
\end{aligned}$$

∎

Another way to write (15) is to say that $\Pr\left[\ln|S^t| - \ln|S^{t+1}| \geq \ln a\right] \leq 3\ell a^{-1}$, which will give the bound (3) on the probability of large jumps when it comes time to apply Theorem 2.

4.2.3 Boundary points

Lemma 5 says that $|S^t|$ seldom drops by too large a ratio at once, but it doesn't tell us much about how quickly $|S^t|$ drops in short hops. To bound this latter quantity, we need to get a bound on how many subranges S^t splinters into through the action of $s(\cdot, \Delta)$. We will do so by showing that only certain points can appear as the boundaries of these subranges in the direction of 0.

For fixed Δ, define for each $i > 0$

$$\beta_i = \left\lceil \frac{\Delta_i + \Delta_{i+1}}{2} \right\rceil$$

and

$$\beta_{-i} = \left\lfloor \frac{\Delta_{-i} + \Delta_{-i-1}}{2} \right\rfloor .$$

Let β be the set of all finite β_i and β_{-i}.

Lemma 6 *Fix S and Δ and let β be defined as above. Suppose that S is positive. Let $M = \{\min(S_{\Delta i\sigma}) : S_{\Delta i\sigma} \neq \emptyset\}$ be the set of minimum elements of subranges $S_{\Delta i\sigma}$ of S. Then M is a subset of S and contains no elements other than*

1. $\min(S)$,
2. Δ_i *for each* $i > 0$,
3. $\Delta_i + 1$ *for each* $i > 0$, *and*
4. *At most one of* β_i *or* $\beta_i + 1$ *for each* $i > 0$,

where the last case holds only with two-sided routing.

If S is negative, the symmetric condition holds for $M = \{\max(S_{\Delta i\sigma}) : S_{\Delta i\sigma} \neq \emptyset\}$.

Proof: Consider some subrange $S_{\Delta i\sigma}$ of S. If $S_{\Delta i\sigma}$ contains $\min(S)$, the first case holds. Otherwise: (a) if $S_{\Delta i\sigma} = S_{\Delta i0}$, the second case holds; (b) if $S_{\Delta i\sigma} = S_{\Delta i+}$, the third case holds; (c) if $S_{\Delta i\sigma} = S_{\Delta i-}$, the fourth case holds, with $\min(S_{\Delta i-}) = \beta_{i-1}$ if $\Delta_{i-1} + \Delta_i$ is odd, and either β_{i-1} or $\beta_{i-1} + 1$ if $\Delta_{i-1} + \Delta_i$ is even, depending on whether the tie-breaking rule assigns β_{i-1} to $S_{\Delta(i-1)+}$ or $S_{\Delta i-}$. ∎

We will call the elements of M *boundary points* of S.

4.2.4 Bounding changes in $\ln|S^t|$

Now we would like to use Lemmas 5 and Lemma 6 to get an upper bound on the rate at which $\ln|S^t|$ drops as a function of the Δ distribution.

The following lemma is used to bound a sum that arises in Lemma 8.

Lemma 7 *Let $c \geq 0$. Let $\sum_{i=1}^{n} x_i = M$ where each $x_i \geq 0$ and at least one x_i is greater than c Let B be the set of all i for which x_i is greater than c. Then*

$$\frac{\sum_{i \in B} x_i \ln x_i}{\sum_{i \in B} x_i} \geq \ln\left(\max\left(c, \frac{M}{n}\right)\right). \tag{16}$$

Proof: If $\frac{M}{n} < c$, we still have $x_i > c$ for all $i \in B$, so the left-hand side cannot be less than $\ln c$. So the interesting case is when $\frac{M}{n} > c$.

Let B have b elements. Then $\sum_{i \notin B} x_i < (n-b)c$ and $\sum_{i \in B} \geq M - (n-b)c = M - nc + bc$. Because $x_i \ln x_i$ is convex, its sum over B is minimized for fixed $\sum_{i \in B} x_i$ by setting all such x_i equal, in which case the left-hand side of (16) becomes simply $\ln(x_i)$ for any $i \in B$.

Now observe that setting all x_i in B equal gives $x_i = \frac{M-nc+bc}{b} = \frac{M-nc}{b} + c \geq \frac{M-nc}{n} + c = \frac{M}{n}$. ∎

Lemma 8 *Fix $a > 1$, and let $S = S^t$ be a positive range with $|S| \geq a$. Define β as in Lemma 6. Let $S' = [\min(S) + \lceil a^{-1}|S| \rceil - 1, \max(S) - 1]$. Then*

$$\begin{aligned} &\mathrm{E}\left[\ln|S^t| - \ln|S^{t+1}| : S^t, \ln|S^t| - \ln|S^{t+1}| < \ln a\right] \\ &\leq \ln \tfrac{1}{1-a^{-1}} + \ln \mathrm{E}[1 + Z : S^t] \end{aligned} \tag{17}$$

where $Z = 2|\Delta \cap S'|$ with one-sided routing and $Z = 2|\Delta \cap S'| + |\beta \cap S'|$ with two-sided routing.

Proof: Call a subrange $S_{\Delta i\sigma}$ *large* if $|S_{\Delta i\sigma}| > a^{-1}|S|$ and *small* otherwise. Observe that $\lceil a^{-1}|S| \rceil \geq 2$, implying any large set has at least two elements.

For any large $S_{\Delta i\sigma}$, $\max(S_{\Delta i\sigma}) \geq \min(S) + \lceil a^{-1}|S| \rceil - 1$. Similarly $\min(S_{\Delta i\sigma}) \leq \max(S) - 1$. So any large $S_{\Delta i\sigma}$ intersects S' in at least one point.

Let $T = \{T_1, T_2, \ldots, T_k\}$ be the set of subranges $S_{\Delta i\sigma}$, large or small, that intersect S'. It is immediate from this definition that $\bigcup T \supseteq S'$ and thus $\sum |T_j| \geq |S'|$.

Using Lemma 6, we can characterize the elements of T as follows:

1. There is at most one set T_j that contains $\min(T_j)$.

2. There is at most one set T_j that has $\min(T_j) = \Delta_i$ for each Δ_i in S'.

3. There is at most one set T_j that has $\min(T_j) = \Delta_i + 1$ for each Δ_i in S'.

4. With two-sided routing, there is at most one set T_j that has $\min(T_j) = \beta_i$ or $\min(T_j) = \beta_i + 1$ for each β_i in S'. Note that there may be a set whose minimum element is $\beta_i + 1$ where $\beta_i = \min(S') - 1$, but this set is already accounted for by the first case.

Thus T has at most $1 + Z = 1 + 2|\Delta \cap S'|$ elements with one-sided routing and at most $1 + Z = 1 + 2|\Delta \cap S'| + |\beta \cap S'|$ elements with two-sided routing.

Conditioning on $|S^{t+1}| > a^{-1}|S|$, $|S^{t+1}|$ is equal to $|S_{\Delta i\sigma}|$ for some large $S_{\Delta i\sigma}$ and thus for some large $T_j \in T$. Which large T_j is chosen is proportional to its size, so for fixed T, we have

$$\begin{aligned} \mathrm{E}[\ln S^{t+1} : T] &= \frac{\sum_{j=1}^{|T|} |T_j| \ln |T_j}{\sum_{j=1}^{|T|} |T_j|} \\ &\geq \ln\left(\max\left(a^{-1}|S|, \frac{|\bigcup T|}{|T|}\right)\right) \\ &\geq \ln\left(\frac{|S'|}{|T|}\right), \end{aligned}$$

where the first inequality follows from Lemma 7.

Now let us compute

$$\begin{aligned}
&\mathrm{E}[\ln|S^t| - \ln|S^{t+1}| : S^t] \\
&\leq \ln|S^t| - \mathrm{E}[\ln|S'| - \ln|T| : S^t] \\
&= \ln\frac{|S^t|}{|S'|} + \mathrm{E}[\ln|T| : S^t] \\
&\leq \ln\frac{1}{1-a^{-1}} + \ln \mathrm{E}[|T| : S^t].
\end{aligned}$$

In the last step we use $\mathrm{E}[\ln|T| : S^t] \leq \ln E[|T| : S^t]$, which follows from the concavity of ln and Jensen's inequality. ∎

4.2.5 Putting the pieces together

We now have all the tools we need to prove our lower bound.

Theorem 9 *Let G be a random graph whose nodes are labeled by the integers. Let Δ_x for each x be a set of integer offsets chosen independently from some common distribution, subject to the constraint that -1 and $+1$ are present in every Δ_x, and let node x have an outgoing edge to $x-\delta$ for each $\delta \in \Delta_x$. Let $\ell = \mathrm{E}[|\Delta|]$. Consider a greedy routing trajectory in G starting at a point chosen uniformly from $1\ldots n$ and ending at 0.*

With one-sided routing, the expected time to reach 0 is

$$\Omega\left(\frac{\ln^2 n}{\ell \ln\ln n}\right) \tag{18}$$

With two-sided routing, the expected time to reach 0 is

$$\Omega\left(\frac{\ln^2 n}{\ell^2 \ln\ln n}\right), \tag{19}$$

provided Δ is generated by including each δ in Δ with probability p_δ, where (a) p is unimodal, (b) p is symmetric about 0, and (c) the choices to include particular δ, δ' are pairwise independent.

Proof: Let $S^0 = \{1\ldots n\}$.

We are going to apply Theorem 2 to the sequence $S^0, S^1, S^2, \ldots$ with $f(S) = \ln|S|$. We have chosen f so that when we reach the target, $f(S) = 0$; so that a lower bound on τ gives a lower bound on the expected time of the routing algorithm. To apply the theorem, we need to show that (a) the probability that $\ln|S|$ drops by a large amount is small, and (b) that the integral in (7) is large.

Let $a = 3\ell\ln^3 n$. By Lemma 5, for all t, $\Pr\left[|S^{t+1}| \leq a^{-1}|S^t|\right] \leq 3\ell a^{-1} = \ln^{-3} n$, and thus $\Pr[\ln|S^t| - \ln|S^{t+1}| \geq \ln a] \leq \ln^{-3} n$. This satisfies (3) with $U = \ln a$ and $\epsilon = \ln^{-3} n$.

For the second step, Theorem 2 requires that we bound the speed of the change in $f(S)$ solely as a function of $f(S)$. For one-sided routing this is not a problem, as Lemma 4 shows that $f(S)$, which reveals $|S|$, characterizes S exactly except when $|S| = 1$ and the lower bound argument is done. For two-sided routing, the situation is more complicated; there may be some S^t which is not of the form $\{1\ldots|S^t|\}$ or $\{0\}$, and we need a bound on the speed at which $\ln|S^t|$ drops that applies equally to all sets of the same size.

It is for this purpose (and only for this purpose) that we use our conditions on Δ for two-sided routing. Suppose that each δ appears in Δ with probability p_δ, that these probabilities are pairwise-independent, and that the sequence p is symmetric and unimodal. Let $\hat{\beta} = \left\{\operatorname{absceil}\left(\frac{x+y}{2}\right) : x, y \in \Delta, x \neq y\right\}$, where $\operatorname{absceil}(z)$, the *absolute ceiling* of z, is $\lceil z \rceil$ when $z \geq 0$ and $\lfloor z \rfloor$ when $z \leq 0$. Observe that $\hat{\beta} \supseteq \beta$; in effect, we are counting in $\hat{\beta}$ all midpoints of pairs of distinct elements of δ without regard to whether the elements are adjacent. For each k, the expected number of distinct pairs x, y with $x + y = z$ and $x, y \in \Delta$ is at most $b_k = \sum_{i=-\infty}^{\infty} p_{k-i} p_i$; this is a convolution of the non-negative, symmetric, and unimodal p sequence with itself and so it is also symmetric and unimodal.It follows that for all $0 \leq k < k'$, $b_k \geq b_{k'}$, and similarly $b_{-k} \geq b_{-k'}$.

Now for the punch line: for each $\delta \neq 0$, $q_\delta = b_{2\delta - \operatorname{sgn} \delta} + b_{2\delta}$ is an upper bound on the expected number of distinct pairs x, y that put δ in β, which is in turn an upper bound on $\Pr[\delta \in \beta]$, and from the unimodularity of b we have that $q_\delta \geq q_{\delta'}$ and $q_{-\delta} \geq q_{-\delta'}$ whenever $0 < \delta < \delta'$. Though q grossly overcounts the elements of β (in particular, it gives a bound on $\mathrm{E}[|\beta|]$ of ℓ^2), its ordering property means that we can bound the expected number of elements of β that appear in some subrange of any positive S^t by using q to bound the expected number of elements that appear in the corresponding subrange of $\{1 \ldots |S^t|\}$, and similarly for negative S^t and $\{-1 \ldots -|S^t|\}$. Because p_i already satisfies a similar ordering property, we can thus bound the number of elements of both Δ and β that hit a fixed subrange of S^t given only $|S^t|$.

Which we now proceed to do. For convenience, formally define $q_i = 0$ for one-sided routing. We will simplify some of the summations by first summing the p_i and q_i over certain pre-defined intervals. For each integer $i > 0$ let $A_i = \{k \in \mathbf{Z} : a^i - 1 \leq k < a^{i+1} - 1\} = \{k \in \mathbf{Z} : \lfloor \ln_a k + 1 \rfloor = i\}$. Let $\gamma_i = \sum_{k \in A_i} 2p_i + q_i$. Note that $\gamma_i \geq 2\,\mathrm{E}[|A_i \cap \Delta|]$ for one-sided routing and $\gamma_i \geq 2\,\mathrm{E}[|A_i \cap \Delta|] + \mathrm{E}[|A_i \cap \beta|]$ for two-sided routing. Observe also that $\sum_{i=0}^{\infty} \gamma_i$ is at most 2ℓ for one-sided routing and by $2\ell + \ell^2$ for two-sided routing.

Consider some $S = S^t$. If $|S| \geq a$, then by Lemma 8 we have

$$\begin{aligned} &\mathrm{E}\left[\ln |S^t| - \ln |S^{t+1}| : S^t, \ln |S^t| - \ln |S^{t+1}| < \ln a\right] \\ &\quad \leq \ln \tfrac{1}{1-a^{-1}} + \ln \mathrm{E}[1 + Z : S^t], \end{aligned} \tag{20}$$

where $Z = 2|\Delta \cap S'|$ with one-sided routing and $Z = 2|\Delta \cap S'| + |\beta \cap S'|$ with two-sided routing, and $S' = [\min(S) + \lceil a^{-1}|S| \rceil - 1, \max(S) - 1]$.

Let us now obtain a bound on $\ln \mathrm{E}[1+Z]$ in terms of $|S|$ and the p_i and q_i. For one-sided routing, we use the fact that $|S| > 1$ implies $S = \{1 \ldots |S|\}$. For two-sided routing, we use monotonicity of the p_i and q_i to replace S with $\{1 \ldots |S|\}$; in particular, to replace a sum of $2p_i + q_i$ over a subrange of S with a sum over subrange of $\{1 \ldots |S|\}$ that is at least as large. In either case, we get that $\ln \mathrm{E}[1 + Z]$ is at most

$$\mu_{\ln |S|} = \ln \frac{1}{1 - a^{-1}} + \ln \left(1 + \sum_{i=\lceil a^{-1}|S| \rceil - 1}^{|S|-1} 2p_i + q_i\right), \tag{21}$$

provided $|S| \geq a$. For $|S| < a$, set $\mu_{\ln |S|} = \ln a$.

Let us now compute m_z, as defined in (6). For $z < \ln a$, $m_z = \ln a$. For larger z, observe that $m_z = \sup\left\{m_{\ln |S|} : e^z \leq |S| < ae^z\right\}$. Now if $e^z \leq |S| < ae^z$, then the bounds on the sum in (21) both lie between $\lceil a^{-1}e^z \rceil - 1$ and $ae^z - 1$, so that

$$\begin{aligned} m_z &\leq \ln \frac{1}{1-a^{-1}} + \ln\left(1 + \sum_{i=\lceil a^{-1}e^z\rceil - 1}^{\lfloor ae^z - 1\rfloor} 2p_i + q_i\right) \\ &\leq \ln \frac{1}{1-a^{-1}} + \ln(1 + \gamma_{z'} + \gamma_{z'+1} + \gamma_{z'+2}), \end{aligned}$$

where $z' = \lfloor z/\ln a\rfloor - 1$.

Finally, compute

$$\begin{aligned} & T(\ln n) \\ &= \int_0^{\ln n} \frac{1}{m_z} dz \\ &\geq \int_{\ln a}^{\ln n} \frac{1}{\ln \frac{1}{1-a^{-1}} + \ln(1 + \gamma_{z'} + \gamma_{z'+1} + \gamma_{z'+2})} dz \\ &\geq \sum_{i=0}^{\lfloor \ln n/\ln a\rfloor - 1} \frac{\ln a}{\ln \frac{1}{1-a^{-1}} + \ln(1 + \gamma_i + \gamma_{i+1} + \gamma_{i+2})}. \end{aligned}$$

To get a lower bound on the sum, note that

$$\sum_{i=0}^{\lfloor \ln n/\ln a\rfloor - 1} (\gamma_i + \gamma_{i+1} + \gamma_{i+2}) \leq 3 \sum_{i=0}^{\lfloor \ln n/\ln a\rfloor + 1} \gamma_i \leq 3 \sum_{i=0}^{\infty} \gamma_i$$

which is at most $L = 6\ell$ for one-sided routing and at most $L = 6\ell + 3\ell^2$ for two-sided routing. In either case, because $\frac{1}{c + \ln(1+x)}$ is convex and decreasing, we have

$$\begin{aligned} & T(\ln n) \\ &\geq \sum_{i=0}^{\lfloor \ln n/\ln a\rfloor - 1} \frac{\ln a}{\ln \frac{1}{1-a^{-1}} + \ln(1 + \gamma_i + \gamma_{i+1} + \gamma_{i+2})} \\ &\geq \sum_{i=0}^{\lfloor \ln n/\ln a\rfloor - 1} \frac{\ln a}{\ln \frac{1}{1-a^{-1}} + \ln\left(1 + \frac{L}{\lfloor \ln n/\ln a\rfloor}\right)} \\ &= \frac{\ln a \lfloor \ln n/\ln a\rfloor}{\ln \frac{1}{1-a^{-1}} + \ln\left(1 + \frac{L}{\lfloor \ln n/\ln a\rfloor}\right)}. \end{aligned} \tag{22}$$

We will now rewrite our bound on $T(\ln n)$ in a more convenient asymptotic form. We will ignore the 1 and concentrate on the large fraction. Recall that $a = 3\ell \ln^3 n$, so $\ln a = \Theta(\ln \ell + \ln\ln n)$. Unless ℓ is polynomial in n, we have $\ln n/\ln a = \omega(1)$ and the numerator simplifies to $\Theta(\ln n)$.

Now let us look at the denominator. Consider first the term $\ln \frac{1}{1-a^{-1}}$. We can rewrite this term as $-\ln(1-a^{-1})$; since a^{-1} goes to zero as ℓ and n grow we have $-\ln(1-a^{-1}) = \Theta(a^{-1}) = \Theta(\ell^{-1}\ln^{-3} n)$. It is unlikely that this term will contribute much.

Turning to the second term, let us use the fact that $\ln(1+x) \leq x$ for $x \geq 0$. Thus

$$\begin{aligned} \ln\left(1 + \frac{L}{\lfloor \ln n/\ln a\rfloor}\right) &\leq \frac{L}{\lfloor \ln n/\ln a\rfloor} \\ &= O\left(\frac{L(\ln l + \ln\ln n)}{\ln n}\right), \end{aligned}$$

and the bound in (22) simplifies to $\Omega\left(\ln^2 n/\left(L(\ln \ell + \ln\ln n)\right)\right)$. We can further assume that $\ell = O(\ln^2 n)$, since otherwise the bound degenerates to $\Omega(1)$, and rewrite it simply as $\Omega\left(\ln^2 n/\left(L \ln\ln n\right)\right)$.

For large L the approximation $\ln(1+x) \leq 1 + \ln x$ for $x \geq 0.59$ is more useful. In this case (22) simplifies to $T(\ln n) = \Omega(\ln n/\ln \ell)$, which has a natural interpretation in terms of the tree of successor nodes of some single starting node.

We are not quite done with Theorem 2 yet, as we still need to plug our T and ϵ into (8) to get a lower bound on $\mathrm{E}[\tau]$. But here we can simply observe that $\epsilon T = O(\ln^{-1} n)$, so the denominator in (8) goes rapidly to 1. Our stated bounds are thus finally obtained by substituting $O(\ell)$ or $O(\ell^2)$ for L. ∎

4.2.6 Possible strengthenings of the lower bound

Examining the proof of Theorem 9, both the ℓ^2 that appears in the bound (19) for two-sided routing and the extra conditions imposed on the Δ distribution arise only as artifacts of our need to project each range S onto $\{1 \ldots |S|\}$ and thus reduce the problem to tracking a single parameter. We believe that a more sophisticated argument that does not collapse ranges together would show a stronger result:

Conjecture 10 *Let G, Δ, and ℓ be as in Theorem 9. Consider a greedy routing trajectory starting at a point chosen uniformly from $1 \ldots n$ and ending at 0.*

Then the expected time to reach 0 is

$$\Omega\left(\frac{\ln^2 n}{\ell \ln\ln n}\right),$$

with either one-sided or two-sided routing, and no constraints on the Δ distribution.

We also believe that the bound continues to hold in higher dimensions than 1. Unfortunately, the fact that we can embed the line in, say, a two-dimensional grid is not enough to justify this belief; divergence to one side or the other of the line may change the distribution of boundaries between segments and break the proof of Theorem 9.

4.3 Upper Bounds

In this section, we present upper bounds on the delivery time of messages in a simple metric space: a one-dimensional real line. To simplify theoretical analysis, the system is set up as follows:

- Nodes are embedded at grid points on the real line.
- Each node u is connected to its nearest neighbor on either side and one or more long-distance neighbors.
- The long-distance neighbors are chosen as per the inverse power-law distribution with exponent 1 i.e. each long-distance neighbor v is chosen with probability inversely proportional to the distance between u and v. Formally, $\Pr[v \text{ is the } i\text{th neighbor of } u] = (\frac{1}{d(u,v)})/(\sum_{v' \neq u} \frac{1}{d(u,v')})$, where $d(u,v)$ is the distance between nodes u and v in the metric space.
- Routing is done greedily by forwarding the message to the neighbor closest to the target node.

We analyze the performance for single and multiple long-distance links in a failure-free network as well as with link and node failures. Note that when we say *node*, we actually refer to a vertex in the virtual overlay network and not a *physical* node as in the earlier sections. Also we use the terms link and edge interchangeably.

4.3.1 Single Long-Distance Link

We first analyze the delivery time in an idealized model with no failures with one long-distance link per node. Kleinberg [6] proved that with n^d nodes embedded at grid points in a d-dimensional grid, with each node u connected to its immediate neighbors and one long-distance neighbor v chosen with probability proportional to $1/d(u,v)^d$, any message can be delivered in time polynomial in $\log n$ using greedy routing. While this result can be directly applied to our model with $d = 1$ to give a $O(\log^2 n)$ delivery time, using Lemma 1, we have a much simpler proof which we give below for completeness.

Theorem 11 *Let each node be connected to its immediate neighbors (at distance 1) and* 1 *long-distance neighbor chosen with probability inversely proportional to its distance from the node. Then the expected delivery time with n nodes in the network is* $T(n) = O(H_n^2)$.

Proof: Let μ_k be the expected number of nodes crossed when the message is at a node which is at a distance k from the destination. Clearly, μ_k is non-decreasing.

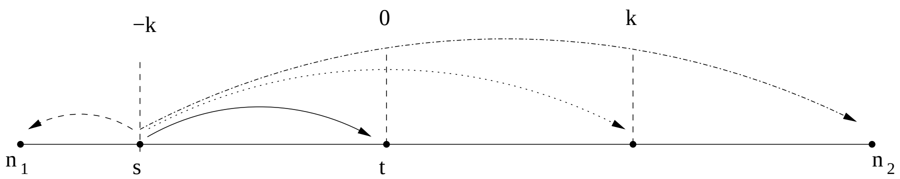


Figure 2: All the possible distances that can be covered from source node s.

Let

$$\mu_k = \frac{\sum_{i=1}^{k} \frac{1}{i} \cdot i}{S} + \frac{\sum_{i=1}^{k-1} \frac{1}{2k-i} \cdot i}{S} + \frac{\sum_{i=1}^{n_1-k} \frac{1}{i} \cdot 1}{S} + \frac{\sum_{i=2k}^{n_2+k} \frac{1}{i} \cdot 1}{S}$$

where

$$S = \sum_{i=1}^{n_1-k} \frac{1}{i} + \sum_{i=1}^{n_2+k} \frac{1}{i} = H_{n_1-k} + H_{n_2+k} < 2H_n$$

Then

$$\mu_k > \frac{1}{S}[k + 0 + H_{n_1-k} + H_{n_2+k} - H_{2k}] > \frac{k}{S} > \frac{k}{2H_n}$$

Clearly, μ_k is non-decreasing, and thus using Lemma 1, we get:

$$T(n) \leq \sum_{k=1}^{n} \frac{1}{\mu_k} = \sum_{k=1}^{n} \frac{2H_n}{k} = O(H_n^2)$$

Thus with this distribution, the delivery time is logarithmic in the number of nodes. ∎

4.3.2 Multiple Long-Distance Links

The next interesting question is whether we can improve this delivery time by using multiple links instead of a single link. In addition to improvement in performance, multiple links also give the added benefit of robustness in the face of failures. We first look at improvement in performance by using multiple links in the system and then go onto analysis of failures in Section 4.3.3. Suppose that there are ℓ links from each node We consider different strategies for generating links and routing depending on number of links ℓ in 2 ranges: $\ell \in [1, \lg n]$ and $\ell \in (\lg n, n^c]$.

In [7], Kleinberg uses a group structure to cover the first case with polylogarithmic links to get $O(\log n)$ delivery time. However, he uses a more complicated algorithm for routing while we obtain the same bound (for the case of a line) using only greedy routing.

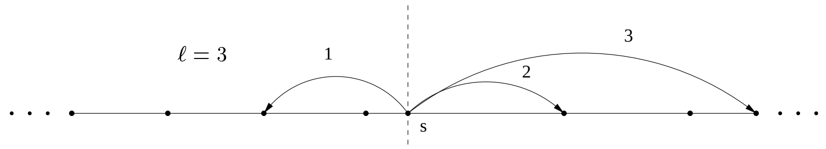


Figure 3: Multiple long-distance links for each node.

4.3.2.1 Upper Bound Let us first consider a randomized strategy for link distribution when $\ell \in [1, \lg n]$.

Theorem 12 *Let each node be connected to its immediate neighbors (at distance 1) and ℓ long-distance neighbors chosen independently with replacement with probability proportional to their distance from the node. Let $\ell \in [1, \lg n]$. Then the expected delivery time $T(n) = O(\lg^2 n/\ell)$.*

Proof: The basic idea for this proof comes from Kleinberg's model [6]. Kleinberg considers a two-dimensional grid with nodes at every grid point. The delivery of the message is divided into phases. A message is said to be in phase j if the distance from the current node to the destination node is between 2^j and 2^{j+1}. There are at most $(\lg n + 1)$ such phases. He proves that the expected time spent in each phase is at most $O(\lg n)$, thus giving a total upper bound on the delivery time as $O(\lg^2 n)$. We use the same phase structure in our model and this proof is on similar lines.

In our multiple-link model, each node has ℓ long-distance neighbors chosen with replacement. The probability that u chooses a node v as its long-distance neighbor is: $1 - (1-q)^\ell$ where $q = \frac{d(u,v)^{-1}}{\sum_{u \neq v} d(u,v)^{-1}}$.

$$\begin{aligned}
1 - (1-q)^\ell &> 1 - (1 - q\ell + \frac{\ell(\ell-1)}{2} q^2) \\
&= q\ell - \frac{\ell(\ell-1)}{2} q^2 = q\ell \left[1 - \frac{(\ell-1)q}{2} \right] \\
&= q\ell \left[1 - \frac{\ell q}{2} + \frac{q}{2} \right] \\
&\geq q\ell \left[1 - \frac{\ell q}{2} \right]
\end{aligned}$$

Notice that $\ell q < 1$, because $q < \frac{1}{\lg n}$ and $\ell \leq \lg n$. So we get probability that u chooses v as its long-distance neighbor is at least

$$q\ell\left[1 - \frac{\ell q}{2}\right] \quad \geq \quad q\ell\left[1 - \frac{1}{2}\right] = \frac{q\ell}{2} = \ell[2d(u,v)H_n]^{-1}.$$

Now suppose that the message is currently in phase j. To end phase j at this step, the message should enter a set of nodes B_j at a distance $\leq 2^j$ of the destination node t. There are at least 2^j nodes in B_j, each within distance $2^{j+1} + 2^j < 2^{j+2}$ of u. So the message enters B_j with probability $\geq 2^j \ell \frac{1}{2H_n 2^{j+2}} = \frac{\ell}{8H_n}$

Let X_j be the total number of steps spent in phase j, then

$$EX_j = \sum_{i=1}^{\infty} Pr[X_j \geq i] \leq \sum_{i=1}^{\infty}\left(1 - \frac{\ell}{8H_n}\right)^{i-1} = \frac{8H_n}{\ell}$$

Now if X denotes the total number of steps, then $X = \sum_{j=0}^{\lg n} X_j$

By linearity of expectation, we get $EX \leq (1 + \lg n)(8H_n/\ell) = O(lg^2 n/\ell)$. ∎

For $\ell \in (\lg n, n^c]$, we use a deterministic strategy. We represent the location of each node as a number to a base $b \geq 2$ and generate links to nodes at distances $1x, 2x, 3x, \ldots, (b-1)x$, for each $x \in \{b^0, b^1, \ldots, b^{\lceil \log_b n \rceil - 1}\}$. Routing is done by eliminating the most significant digit of the distance at each step. As this distance can be at most $b^{\lceil \log_b n \rceil}$, we get $T(n) = O(\log_b n)$. This strategy is similar in spirit to Plaxton's algorithm [11].

For $\ell = O(\lg n)$, each node is linked to nodes at distances $1, 2, 4, 8, \ldots, 2^{\lg n - 1}$, giving $T(n) = O(\lg n)$. Also, if $\ell = O(\sqrt{n})$, links are to nodes at distances $1, 2, 3, \ldots, \sqrt{n}, 2\sqrt{n}, 3\sqrt{n}, \ldots, \sqrt{n}(\sqrt{n} - 1)$. giving $T(n) = O(1)$. In fact, $T(n) = O(1)$ for $b = n^c$, for any fixed c.

Theorem 13 *Choose an integer $b > 1$. With $\ell = (b-1)\lceil \log_b n \rceil$, let each node link to nodes at distances $1x, 2x, 3x, \ldots, (b-1)x$, for each $x \in \{b^0, b^1, \ldots, b^{\lceil \log_b n \rceil - 1}\}$. Then the delivery time $T(n) = O(\log_b n)$.*

Proof: Let $d_1, d_2, \ldots d_t$ be the distances of the successive nodes in the delivery path from the target t; d_1 being the distance of the source node and $d_t = 0$. For each $d_i, \exists k_i \in \{0, 1, \ldots, \lfloor \log_b n \rfloor\}$

$$b^{k_i} \leq d_i < b^{k_i + 1}$$

$$1 \leq \lfloor \frac{d_i}{b^{k_i}} \rfloor < b$$

Now each node is connected to the node at distance $b^{k_i} \lfloor \frac{d_i}{b^{k_i}} \rfloor$. So

$$d_{i+1} = d_i - b^{k_i} \lfloor \frac{d_i}{b^{k_i}} \rfloor = d_i \mod b^{k_i} < b^{k_i}$$

Thus k_i drops by at least 1 at every step. As $k_1 \leq \lceil \log_b n \rceil$, we get $T(n) = O(\log_b n)$. ∎

4.3.2.2 Lower Bound

We now look at the lower bound for $\ell \in (\lg n, n^c]$.

Theorem 14 *Let $\ell \in (\lg n, n^c]$. Then for any link distribution, the delivery time $T(n) = \Omega(\frac{\lg n}{\lg \ell})$.*

Proof: With ℓ links for each node, we can reach at most ℓ^k nodes at step k. Assuming that the minimum time to reach all n nodes is T, $\ell^T = n$. This gives a lower bound on $T = \Omega(\frac{\lg n}{\lg \ell})$. ∎

So, $T = \Omega(1)$ for $\ell = n^c$ and $T = \Omega(\frac{\lg n}{\lg \lg n})$ for $\ell = \lg n$.

4.3.3 Failure of Links

It appears that our linking strategies may fail to give the same delivery time in case the links fail. However, we show that we get reasonable performance even with link failures. In our model, we assume that each link is present independently with probability p. Let us first look at the randomized strategy for number of links $\ell \in [1, \lg n]$.

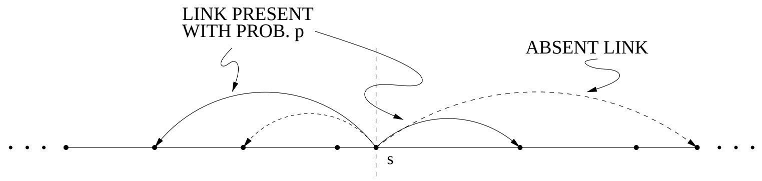


Figure 4: Each long-distance link is present with probability p.

For $\ell \in [1, \lg n]$, our proof is on similar lines as our proof for the case with no failures. Intuitively, since some of the links fail, we expect to spend more time in each phase and this time should be inversely proportional to the probability with which the links are present. We prove that the expected time spent in one phase is $O(\lg n/p\ell)$, which gives a total delivery time of $O(\lg^2 n/p\ell)$. We assume that the links to the immediate neighbors are always present so that a message is always delivered even if it takes very long for that.

Theorem 15 *Let the model be as in Theorem 12. Assume that the links to the immediate neighbors are always present. If the probability of a long-distance link being present is p, then the expected delivery time is* $O(\lg^2 n/p\ell)$.

Proof: The proof is on the same lines as the proof for Theorem 12. Recall that in case of no link failures, the probability that u chooses a node v as its long-distance neighbor is at least $q\ell/2$ where $q = \frac{d(u,v)^{-1}}{\sum_{u \neq v} d(u,v)^{-1}}$.

Now when we consider link failures, given that u chose v as its long-distance neighbor, the probability that there is a link present between u and v is p. So, the probability that u chooses a node v as its long-distance neighbor is at least $pq\ell/2 = p\ell[2d(u,v)H_n]^{-1}$.

The rest of the proof is the same as the proof for theorem 12. Let X_j be the number of steps spent in phase j.

$$EX_j = \sum_{i=1}^{\infty} Pr[X_j \geq i] = \frac{8H_n}{p\ell}$$

If X denotes the total number of steps, then by linearity of expectation, we get $EX \leq (1 + \lg n)(8H_n/p\ell) = O(lg^2 n/p\ell)$. ∎

We turn to the deterministic strategy with $\ell \in (\lg n, n^c]$ links. A similar intuition works for $\ell \in (\lg n, n^c]$. If a link fails, then the node has to take a shorter long-distance link, which will not take the message as close to the target as the initial failed link. Clearly as p decreases, the message has to take shorter and shorter links which increases the delivery time.

To make the analysis simpler, we change the link model a bit and let each node be connected to other nodes at distances $b^0, b^1, b^2, \ldots, b^{\lfloor log_b n \rfloor}$. Once again, we compute the expected distance covered from the current node and use Lemma 1 to get a delivery time of $O(b \lg n/p)$. Clearly as p

decreases, the delivery time increases whereas as b increases, the information stored at each node decreases and hence the delivery time increases.

Theorem 16 *Let the number of links be $O(\log_b n)$, and let each node have a link to distances $b^0, b^1, b^2, \ldots, b^{\lfloor log_b n \rfloor}$. Assume that the links to the nearest neighbors are always present. If the probability of a link being present is p, then the delivery time $T(n) = O(bH_n/p)$.*

Proof: Let the distance of the current node from the destination be k. Let μ_k represent the distance covered starting from this node. Then with probability p, there will be a link covering $b^{\lfloor \log_b k \rfloor}$ distance. If this link is absent with probability $q = 1 - p$, then we can cover a distance $b^{\lfloor \log_b k \rfloor - 1}$ with a single link with probability pq and so on. In general, the average distance μ_k covered when the message is at distance k from the destination is

$$\begin{aligned}
\mu_k &= pb^{\lfloor \log_b k \rfloor} + pqb^{\lfloor \log_b k \rfloor - 1} + \ldots + pq^{\lfloor \log_b k \rfloor - 1} b^1 + q^{\lfloor \log_b k \rfloor} b^0 \\
&\geq \sum_{i=0}^{\lfloor \log_b k \rfloor} pb^{\lfloor \log_b k \rfloor - i} q^i \\
&= pb^{\lfloor \log_b k \rfloor} \sum_{i=0}^{\lfloor \log_b k \rfloor} \left(\frac{q}{b}\right)^i \\
&= pb^{\lfloor \log_b k \rfloor} \frac{1 - (q/b)^{\lfloor \log_b k \rfloor + 1}}{1 - (q/b)} \\
&= \frac{p(b^{\lfloor \log_b k \rfloor + 1} - q^{\lfloor \log_b k \rfloor + 1})}{b - q} \\
&\geq \frac{p(bk/b - 1)}{b - q} \\
&\geq \frac{p(k-1)}{2(b-q)}
\end{aligned}$$

Using Lemma 1, we get

$$T(n) \leq \sum_{k=1}^{n} \frac{1}{\mu_k} = 1 + \sum_{k=2}^{n} \frac{2(b-q)}{p(k-1)} = 1 + \frac{2(b-q)}{p} \left[\sum_{k=2}^{n} \frac{1}{(k-1)} \right] = O(bH_n/p)$$

As p decreases, the delivery time increases. Since this model has been simplified for analysis, as b decreases we get more links ($\log_b n$) and so the delivery time decreases. ∎

4.3.4 Failure of Nodes

We can consider two different situations for effect of node failures on system performance. In the first case as described in Section 4.3.4.1, some of the nodes may fail and then the remaining nodes will link to each other as per the link distribution. On the other hand, as explained in Section 4.3.4.2, the nodes first link to their neighbors and then some of the nodes may fail.

4.3.4.1 Binomially Distributed Nodes We note that we get the same upper bound even if we assume that the nodes are binomially distributed on the grid points of a line. Let p be the probability that a node is present at any point. Here also, each node is connected to its nearest neighbors and one long-distance neighbor. In addition, the probability of choosing a particular node as a long-distance neighbor is conditioned on the existence of that node.

Theorem 17 *Let the model be as in Theorem 11. Let each node be present with probability p and all nodes link only to existing nodes. Then the expected delivery time $T(n) = O(\lg^2 n)$.*

Proof: The expected drop μ_k is given as follows:

$$\begin{aligned}
\mu_k &= \frac{\sum_{i=1}^{k} \frac{1}{i} \cdot i \cdot p}{p \cdot S} + \frac{\sum_{i=1}^{k-1} \frac{1}{2k-i} \cdot i \cdot p}{p \cdot S} + \frac{\sum_{i=1}^{n_1-k} \frac{1}{i} \cdot 1 \cdot p}{p \cdot S} + \frac{\sum_{i=2k}^{n_2+k} \frac{1}{i} \cdot 1 \cdot p}{p \cdot S} \\
&> \frac{1}{S}[k + 0 + H_{n_1-k} + H_{n_2+k} - H_{2k}] \\
&> \frac{k}{S} > \frac{k}{2H_n}
\end{aligned}$$

Using Lemma 1, we get $T(n) \leq \sum_{k=1}^{n} 1/\mu_k = O(H_n^2)$. This is exactly the same result that we get in the Section 4.3.1 where all the nodes are present. ∎

This result is not surprising because if nodes link only to other existing nodes, the only difference is that we get a smaller random graph. This does not affect the routing algorithm or the delivery time.

4.3.4.2 General Failures We observe that the analysis for node failure is not as simple as that for link failures because we no longer have the important property of independence that we have in latter case. In case of link failures, the nodes first choose their neighbors and then it is possible that some of these links fail. Here, the event that a node is connected to another node is completely independent of the event that, say, its neighbor is connected to the same node. Each link fails independently and so the reachability from different nodes to a given target depends only on the presence of the link between the two nodes in question.

In case of node failure, this important independence property is no longer true. Suppose that a node u cannot communicate with some other node v (because v failed), even though there may be a functional link between u and v. Now the probability of some other node w being able to communicate with v is not independent of the probability that u can communicate with v because the probability of v being absent is common for both the cases. This complicates the analysis of the performance because it is not longer the case that if one node cannot communicate with some other node, it has a good chance of doing so by passing the message to its neighbor.

In order to analyze this situation, we consider jumps only to one lower phase rather than jumping over several phases. The idea is that the jumps between phases are independent so once we move from phase j to phase $j-1$, further routing no longer depends on any nodes in phase j. We can condition on the number of nodes being alive in the lower phase and estimate the time spent in each phase. Intuitively, if a node is present with probability p, we would expect to wait for a time inversely proportional to p in anticipation of finding a node in the lower phase to jump to.

Theorem 18 *Let the model be as in Theorem 12 and let each node fail with probability p. Then the expected delivery time is* $O(\lg^2 n/(1-p))$.

Proof: Let T be the time taken to drop down from layer j to layer $j-1$. Let l out of N nodes be alive in layer $j-1$ and let q be the probability that a node in layer j is connected to some node in layer $j-1$. Then the expected time to drop to layer $j-1$, given that there are l live nodes in it, is given by

$$\begin{aligned} E[T|l] &= 1 + \left[(1-q) + \frac{q(N-l)}{N}\right] E[T|l] \\ &= \frac{N}{ql} \end{aligned}$$

Now l can vary between 1 and N. (Note that here l cannot be 0 because if there are no alive nodes in the lower layer, the routing totally fails at this point) p is the probability that a node in the lower layer fails. We get

$$\begin{aligned} E[T] &= \sum_{l=1}^{N} \frac{N}{ql}\left[p^{N-l}(1-p)^l \binom{N}{l}\right] \\ &= \frac{N}{q}\sum_{l=1}^{N} \frac{1}{l} p^{N-l}(1-p)^l \binom{N}{l} \\ &\leq \frac{N}{q}\sum_{l=1}^{N} \frac{2}{l+1} p^{N-l}(1-p)^l \binom{N}{l} \\ &= \frac{2N}{q(N+1)(1-p)} \sum_{l=1}^{N} p^{N-l}(1-p)^{l+1} \binom{N+1}{l+1} \\ &\leq \frac{2N}{q(N+1)(1-p)} \left[p + (1-p)\right]^{N+1} \\ &= \frac{2N}{q(N+1)(1-p)} \end{aligned}$$

As expected, we see that the expected waiting time in a layer is inversely proportional to the probability of being connected to a node in the lower layer and the probability of such a node being alive.

For our randomized routing strategy with $[1, \lg n]$ links, $q \approx 1/(H_n \ell)$ and there are at most $(\lg n + 1)$ layers, giving an expected delivery time of $O(\lg^2 n/(1-p)\ell)$. ∎

In contrast, for our *deterministic* routing strategy, certain carefully chosen node failures can lead to dismal situations where a message can get stuck in a local neighborhood with no hope of getting out of it or eventually reaching the destination node. We conjecture that this should be a very low probability event and hence its occurrence will not affect the delivery time considerably. We have not yet analyzed this siutation formally.

5 Construction of Graphs

As the group of nodes present in the network changes, so does the graph of the virtual overlay network. In order for our routing techniques to be effective, the graph must always exhibit the property that the likelihood of any two vertices v, u being connected is $\Omega(d(v,u)^{-1})$. We describe briefly a heuristic approach to construct and maintain a graph with such an invariant.

Since the choice of edges leaving each vertex is independent of the choices of other vertices, we can assume that points in the metric space are added one at a time. Let v be the k-th point to be added. Point v chooses the sinks of its outgoing edges according to the inverse-distance distribution and connects to them by running the search algorithm. If a desired sink u is not present, v connects to u's closest, present neighbor. In effect, each of the $k-1$ points already present before v is surrounded by a basin of attraction, collecting probability mass in proportion to its length. Since we assume the hash function populates the metric space evenly, and because of absolute symmetry, the basin length L has the same distribution for all points. It is easy to see that with high probability, L will not be much smaller than its expectation: $\mathrm{Prob}(L \leq c \cdot k^{-1}) = 1 - (1 - c \cdot k^{-1})^{k-1}$. A lower bound on the probability that the edge (v, u) is present is $c' \cdot k^{-1} \cdot d(v, f)^{-1}$, where f is the point in u's basin that is the farthest from v.[4] However, the bound holds only if u is amongst the $k-1$ points added before v. Otherwise, the aforementioned probability is 0, which means that we need to amend our linking strategy so as to transfer probability mass from the former case to the latter one. We describe next how to accomplish this task.

Let u be a new point. We give earlier points the opportunity to obtain outgoing edges to u by having u (1) calculate the number of incoming edges it "should" have from points added before it arrived, and (2) choose such points according to an appropriate distribution α.[5] If ℓ is the number of outgoing edges for each point, then ℓ will also be the expected number of incoming edges that u has to estimate in step (1). For graphs with a large number of points n, each point has roughly $1/n$ chance of ending at u because of symmetry. The number of links ending at u is thus distributed according to a Poisson distribution with rate ℓ, that is, the probability that u has k incoming edges is $\frac{e^{-l} l^k}{k!}$, and the expectation of the distribution is ℓ. After step (2) is completed by u, each chosen point v responds to u's request by using a distribution β to choose one of its existing outgoing edges to replace with an edge to u. It is easy to show that if α and β are proportional to the inverse of the distance between points, the resulting graph has the property we want. This procedure can be repeated to allow for regeneration of links when a network node crashes.

We believe that our method will give a distribution close enough to the desired distribution for the routing algorithm to work. Unfortunately, the construction process interacts in a complicated way with the routing algorithm, which makes analysis difficult. We are currently conducting simulations to assess the effectiveness of constructing a graph using our heuristic.

There has recently been related work [10] on how to construct, with the support of a central server, random graphs with many desirable properties, such as small diameter and guaranteed connectivity with high probability. Although it is not clear what kind of fault-tolerance properties this approach offers if the central server crashes, or how the constructed graph can be used for efficient routing, it is likely that similar techniques could be useful in our setting.

6 Experimental Results

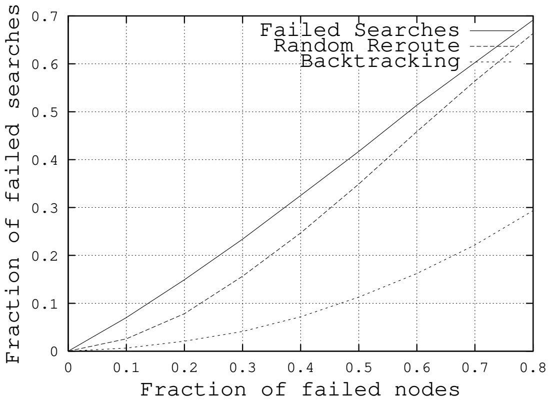
We simulated a network at the *application level* with $n = 2^{17}$ nodes. Each node has $\lg n = 17$ links chosen using the randomized rule explained in Section 4.3. Routing is done greedily by forwarding a message to the neighbor closest to its target node. In each simulation, the network is set up afresh and a fraction p of the nodes fail. We choose random source and destination nodes which

[4] The constant c' has absorbed c and the normalizing constant for the distribution.

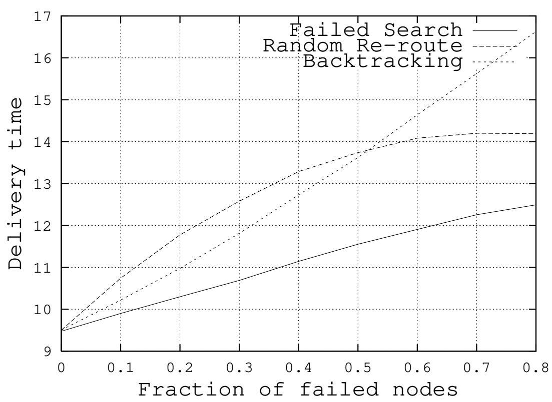
[5] All this can be easily calculated by u since the link probabilities are symmetric.

have not failed and route a message between them. For each value of p, we ran 1000 simulations delivering 100 messages in each simulation, and averaged the number of delivery hops for successful searches and number of failed searches.

With node failures, a node may not be able to find a live neighbor that is closer to the target node than itself. We studied three possible strategies to overcome this problem as follows: (i) Terminate the search. (ii) Randomly choose another node, deliver the message to this new node and then try to deliver the message from this node to the original destination node (similar to the hypercube routing strategy as explained in [14]). (iii) Keep track of a fixed number (in our simulations, 5) of nodes through which the message is last routed and backtrack. When the search reaches a node from where it cannot proceed, it backtracks to the most recently visited node from this list and chooses the next best neighbor to route to. For all these strategies we note that once a node chooses its best neighbor, it does not send the message to any other link if it finds out that the best neighbor has failed.



(a) Fraction of failed searches



(b) Delivery Time

Figure 5: (a) The fraction of messages that fail to be delivered as a function of the fraction of failed nodes. (b) The delivery time for successful searches as a function of the fraction of failed nodes.

Figure 5 shows the fraction of messages that fail to be delivered and the number of hops for successful searches versus the fraction of failed nodes. It is very interesting to see how well the system behaves even with such a large number of failed nodes. In addition, backtracking gives a significant improvement in reducing the number of failures as compared to the other two methods, although it may take a longer time for delivery.

Our results may not be directly comparable to those of CAN[12] and Chord[13] since they use different simulators for their results. However, we see that we get results as good as theirs. Even if we just terminate the search, we get less than p fraction of failed searches with p fraction of failed nodes. Chord[13] has roughly the same performance *after* their network stabilizes from some repair mechanism. Further, with backtracking we see that with 80% failed nodes, we still get less than 30% failed searches. These results are very promising and it would be very interesting to study backtracking analytically.

Model	Number of Links ℓ	Upper Bound	Lower Bound
No failures	1	$O(\lg^2 n)$	$\Omega(\frac{\ln^2 n}{\ln \ln n})$
	$[1, \lg n]$	$O(\frac{lg^2 n}{\ell})$	$\Omega(\frac{\ln^2 n}{\ell \ln \ln n})$
	$[\lg n, n^c]$	$O(\frac{\lg n}{\lg b})$	$\Omega(\frac{\lg n}{\lg \ell})$
Pr[Link present]=p	$[1, \lg n]$	$O(\frac{\lg^2 n}{p\ell})$	-
	$[\lg n, n^c]$	$O(\frac{b \lg n}{p})$	-
Pr[Node present]=p	$[1, \lg n]$	$O(\frac{\lg^2 n}{p\ell})$	-

7 Conclusions and Future Work

The following table summarizes our upper and lower bounds[6]:

We have shown that greedy routing in an overlay network organized as a random graph in a metric space can be a nearly optimal mechanism for searching a peer-to-peer system with low message complexity, even in the presence of many faults. We see this as an important first step in the design of efficient algorithms for such networks, but many issues still need to be addressed. Our results mostly apply to one-dimensional metric spaces like the line or a circle. One interesting possibility is whether similar strategies would work for higher-dimensional spaces, particularly ones in which some of the dimensions represent the actual physical distribution of the nodes in real space; good network-building and search mechanisms for this model might allow efficient location of nearby instances of a resource without having to resort to local flooding (as in [5]). Another promising direction would be to study the security properties of greedy routing schemes, to see how they can be adapted to provide desirable properties like anonymity or robustness against Byzantine failures.

References

[1] FREENET. http://www.freenet.sourceforge.net.

[2] GNUTELLA. http://gnutella.wego.com.

[3] A. D. Joseph, J. Kubiatowicz, and B. Y. Zhao. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California, Berkeley, Apr 2001.

[4] R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988.

[5] D. Kempe, J. M. Kleinberg, and A. J. Demers. Spatial gossip and resource location protocols. In *Proceedings of 33rd Annual ACM Symposium on Theory of Computing*, pages 163–172, 2001.

[6] In the upper bound with $(\lg n, n^c]$ links, the number of links $\ell = O(b \log_b n)$. Also, the deterministic strategy used for links $\ell \in (\lg n, n^c]$, with link failures is slightly different that the one with no failures, and $\ell = O(\log_b n)$. In the lower bound column, the bound for $[1, \lg n]$ links is for one-sided routing.

[6] J. Kleinberg. The small-world phenomenon:an algorithmic perspective. Technical Report 99-1776, Cornell University, Oct. 1999.

[7] J. Kleinberg. Small-world phenomena and the dynamics of information. 2001.

[8] MORPHEUS. http://www.musiccity.com.

[9] NAPSTER. http://www.napster.com.

[10] G. Panduragan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. In *Proceedings of 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2001.

[11] C. Plaxton, R. Rajaram, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, June 1997.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM*, pages 161–170, 2001.

[13] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishna. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM 2001*, pages 149–160, 2001.

[14] L. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11:350–361, 1982.