Greedy Routing in Peer-to-Peer Systems*

James Aspnes† Zoë Diamadi‡ Gauri Shah§

June 22, 2006

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 [10], is close to optimal. We also give efficient heuristics to dynamically maintain such a system as new nodes arrive and old nodes depart. Finally, we give 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 a critical feature of such a system as any central server not only provides a vulnerable point of failure but also does not take advantage of 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

*This is an extended version of the paper *Fault-tolerant Routing in Peer-to-Peer Systems* 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.

‡McKinsey & Company, Three Landmark Square, Stamford, CT 06901. Email: zoe.diamadi@mckinsey.com. Much of the work described in this paper was performed at Yale University. Supported in part by ONR grant N00014-01-1-0795.

§Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043. Email: gauri.shah@aya.yale.edu. Much of the work described in this paper was performed at Yale University. Supported by NSF grants CCR-9820888 and CCR-0098078.

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.

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 an 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 in any case the space requirements are small. The information stored at a node consists only of a network address for each neighbor.

The rest of the paper is organized as follows. Section 2 explains our abstract model in detail, and Section 3 describes some existing peer-to-peer systems. We prove our results for routing in Section 4. In Section 5, we present a heuristic method for constructing the random graph and provide experimental results that show its performance in practice. Section 6 describes results of experiments we performed to test the routing performance of our constructed distributed data structure. Conclusions and future work are discussed 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 a random graph by appropriately linking these points, and (3) efficiently locate resources by routing greedily along the edges of the graph. 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 assume a hash function $h : K \to V$ such that resource r maps to the point $v = h(key(r))$ in a metric space (V, d), where V is the point set and d is the distance metric as shown in Figure 1. The hash function is assumed to populate the metric space evenly. 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 from the points embedded in V. We assume that each newly-arrived node n is initially connected to some other node in N. Each node n generates the outgoing links for each vertex $v \in V_n$ independently. A link $(v, u) \in V_n \times V_m$ 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. Node n constructs each link by executing the search algorithm to locate the resource that is mapped to the sink of that link. If the metric space is not populated densely enough, the choice of a sink may result in a vertex corresponding to an absent resource.

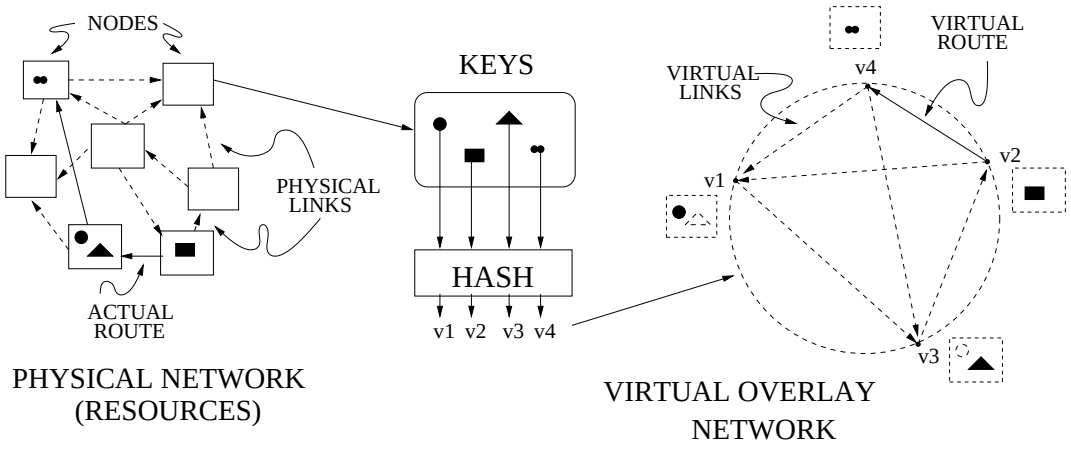


Figure 1: An example of the metric-space embedding.

In that case, n chooses the neighbor present closest to the original sink. Moving to nearby vertices will introduce some bias in the link distribution, but the magnitude of 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. 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 itself.[1] Each node needs only local information, namely its set of neighbors in I^t, to participate in the resource location. Routing is done greedily by forwarding the message 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 [10], in which points in a two-dimensional grid are connected by links 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 links leaving each node. Getting good performance using his technique depends both on having a complete two-dimensional grid of nodes and on very carefully adjusting the exponent of the random link 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 links (due to failure of one of

[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}, ..., I^{t_k}.

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 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.

3 Related work

Most of the peer-to-peer systems in widespread use are not scalable. Napster [15] 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 not being scalable. Gnutella [6] floods the network to locate a resource. Flooding creates a trade-off between overloading every node in the network for each request and cutting off searches before completion. While the use of super-peers [13] ameliorates the problem somewhat in practice, it does not improve performance in the limit.

Some of these first-generation systems have inspired the development of more sophisticated ones like CAN [18], Chord [20] and Tapestry [23]. CAN partitions a d-dimensional metric space into *zones*. Each key is mapped to a point in some zone and stored at the node that owns the zone. Each node stores $O(d)$ information, and resource location, done by greedy routing, takes $O(dn^{1/d})$ time. Chord maps nodes to identities of m bits placed around a *modulo* 2^m *identifier circle*. Resources are stored at existing *successor* nodes of the nodes they are mapped to. Each node 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 $O(\log n)$ delivery time with n nodes in the system. Tapestry uses Plaxton's algorithm [17], a form of suffix-based, hypercube routing, as the routing mechanism: in this algorithm, the message is forwarded deterministically to a node whose identifier is one digit closer to the target identifier. To this end, each node maintains $O(\log n)$ pieces of information and delivery time is also $O(\log n)$.

Although these systems seem vastly different, there is a recurrent underlying theme in the use of some variant of an overlay metric space in which the nodes are embedded. The location of a resource in this metric space is determined by its key. Each node maintains some information about its neighbors in the metric space, and routing is then simply done by forwarding packets to neighbors *closer* to the target node with respect to the metric. 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. In Chord, the nodes can be thought of being embedded on grid points on a real circle, with distances measured along the circumference of the circle providing 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.

The main contribution of this paper is twofold. We explain why most of these systems achieve similar performance guarantees for greedy routing by describing a general setting for such overlay metric spaces, although most of our results apply only in one-dimensional spaces. Perhaps more importantly, we show why greedy routing might not be the best way to perform resource location in peer-to-peer systems by providing lower bounds on the message complexity of locating particular resources in a graph with nodes linked according to a fixed distribution. Indeed, second generation peer-to-peer systems like Viceroy [12], Koorde [7], and the distance halving DHT of [14] do not use greedy routing or fixed link distributions. As a result, they meet or beat the lower bounds presented in this paper.

Barrière *et al.*[1] reached independently from this work one of the upper bounds presented here (Theorem 12). While this upper bound was first presented by Kleinberg in [10], we use simpler methodology from that in [10] and [1] to derive it. Furthermore, our results are orthogonal to those in [1]. The authors of that work concentrate on bounds for inverse power-law distributions with different exponents, for systems with one long-distance link and no failures. We establish bounds for the power-law distribution with exponent 1 under different scenarios: for single or multiple long-distance links, under no failures, or in the presence of node or link failures. Finally, with Theorem 10 we answer an open problem posed in [1], namely whether their lower bound of $\Omega(\log^2 n)$ for greedy routing with power-law distributions extends to all distance-invariant distributions.

Similar models to the one presented here have been discussed before in the context of percolation models, e.g., [2, 3, 4, 5]. They study random graphs embedded in a metric space $\mathbb{Z}^d$ in the cases where the probability of an edge between two graph vertices falls off with exponent s and focus on providing bounds for the graph diameter for different ranges of s in respect to d. In these papers, it is shown that short paths in such random graphs exist, but as we show with this work, greedy routing fails to find them.

4 Routing

In this section, we present our lower and upper bounds on routing. We consider greedy routing in a graph embedded in a line where each node is connected to its immediate neighbors and to multiple long-distance neighbors chosen according to a fixed link distribution. We give lower bounds for greedy routing for *any* link distribution satisfying certain properties (Theorem 10). We also present upper bounds in the same model where the long-distance links are chosen as per the inverse power-law distribution with exponent 1 and analyze the effects on performance in the presence of failures.

For both the upper and lower bounds we model the path taken by a search through the system as a Markov chain. We can do so because (a) the outgoing links from each node are chosen independently of the outgoing links from all other nodes, and (b) no node ever appears in the same search path twice; these two facts together mean that we can treat each node that appears in the search path as newly generated, and treat each step in the routing protocol as generated by a new random choice of outgoing links from the current node. Condition (a) is assumed in our basic model; condition (b) depends on the search always making progress, which we guarantee by requiring that each node is always connected to its immediate neighbors.

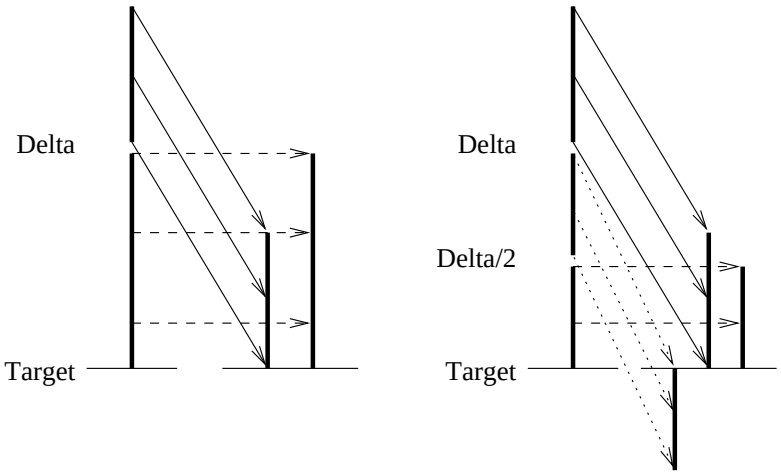


Figure 2: One step of the aggregate chain after choosing a single link distance δ and either one-sided routing (left picture) or two-sided routing (right picture). Tokens starting at distance δ or more from the target drop by δ in either case. Tokens with initial distance between $\delta/2$ and δ drop by δ in the two-sided case (because this moves them closer to the target), but stay where they are in the one-sided case. Closer tokens do not move in either case. In both cases, applying the length-δ link splits the original range into two or more ranges, each of which is chosen with probability proportional to its length.

The upper bounds can then be obtained by choosing particular link distributions and analyzing the trajectories of the resulting Markov chain, whose expected lengths we bound using a well-known bound on probabilistic recurrence relations due to Karp *et al.*[8].

The lower bounds are more difficult. Two issues arise: first, we must consider a wider variety of possible link distributions, and cannot simply analyze a single Markov chain. Second, we can no longer assume a fixed (but arbitrary) starting location as in the upper bound argument: if the starting location and target are fixed, the designer of the link distribution can simply arrange for there to be a direct link from the start to the finish with high probability.

The second issue is handled by assuming that the initial location is chosen uniformly at random from some range of nodes at integer positions 1 to n, with the target at position 0. This yields n parallel Markov processes, one for each of the starting points, and we track the trajectories of these parallel Markov processes simultaneously, under a coupling assumption where the outgoing links from the current node in each of these parallel processes have the same offsets. We show that this assumption gives rise to a new aggregate Markov process whose states are ranges of nodes, where the initial uniform distribution over nodes in the range 1 to n evolves at each step to a probabilistic mixture over smaller ranges of nodes (see Figure 2). By tracking the maximum element of the current range, we can detect when the target is reached, because the target is reached only when this quantity hits zero.

To bound the time taken to reduce the maximum element to zero, we use an upside-down version of the Karp *et al.* inequality for probabilistic recurrence relations, which we prove in Section 4.1. The intuition behind the Karp *et al.* bound is that in a probabilistic recurrence relation of the form $T(n) = 1 + T(n - X)$, where X is a random variable, the expected value of X conditioned on n acts like a "speed" at which the process passes n, and that the total time to reach 0 can be

bounded above by integrating the inverse of this speed, subject to a monotonicity condition that essentially shows that the speed at which the process passes some point n is at least the expected drop from n, even if the process passes n starting from some larger point n'. We show that a similar integral gives a lower bound on the time, for a suitable definition of the expected speed at each point; this result is presented as Theorem 2. Our theorem does not require monotonicity; instead, the "instantaneous speed" at each point is defined as the supremum of the expected drop from an appropriate set of higher points.

Our lower bounds are then obtained by applying Theorem 2 to the *logarithm* of the maximum element of the current range in the parallel-trajectories Markov process described earlier. The intuition is that only links that jump distances within a small factor of the distance between the current node n and the target will yield much progress in the logarithm. For one-sided routing, where the search path cannot go past the target, a long link that jumps more than n units will not be used; but a short link that jumps much fewer than n units will reduce $\lg n$ by only a small amount. In order to make speedy progress, the link distribution must supply a link whose length is close to n; but it must also continue to do so as n shrinks, to keep the process from getting stuck later. Our proof shows that the best choice is to assign equal weight to each range of offsets *after* taking their logarithms; this gives precisely the $\Theta(1/d)$ distribution originally used by Kleinberg [10].

This section is organized as follows. Section 4.1 describes the Karp *et al.* upper bound on probabilistic recurrence relations, and gives our new lower bound. Section 4.2 contains the lower bound argument. Section 4.3 gives upper bounds for various link distributions and failure patterns.

4.1 Probabilistic recurrence relations

Some of our upper bounds will be proved using a well-known upper bound of Karp *et al.*[8] 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 ([8]) *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 (9).

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*

$$\lim_{t\to\infty} \Pr[f(X_t) = 0] = 1 \tag{2}$$

and, for all t,

$$\Pr[f(X_t) - f(X_{t+1}) \geq 0 : X_t] = 1. \tag{3}$$

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

$$\Pr[f(X_t) - f(X_{t+1}) \geq U : X_t = x] \leq \epsilon. \tag{4}$$

Let

$$\tau = \min\{t : f(X_t) = 0\}. \tag{5}$$

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

$$\mu_x \geq \mathrm{E}[f(X_t) - f(X_{t+1}) : X_t = x, f(X_t) - f(X_{t+1}) < U]. \tag{6}$$

Now define

$$m_z = \sup\{\mu_x : x \in S, f(x) \in [z, z+U)\}, \tag{7}$$

and define

$$T(x) = \int_0^{f(x)} \frac{1}{m_z} dz. \tag{8}$$

Then

$$\mathrm{E}[\tau] \geq \frac{T(X_0)}{\epsilon T(X_0) + (1-\epsilon)}. \tag{9}$$

Proof: Define

$$Y_t = \begin{cases} T(X_t) & \text{if } f(X_{t'}) - f(X_{t'+1}) < U \text{ for all } t' < t, \text{ or} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The idea is that Y_t drops to zero immediately if a long jump occurs. We will show that even with such overeager jumping, Y_t does not drop too quickly on average. 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 $f(X_t) - f(X_{t+1}) < U$, that is, that the jump from $f(X_t)$ to $f(X_{t+1})$ is a short jump. Now compute

$$\begin{aligned}
E\left[Y_t - Y_{t+1} : \mathcal{F}_t\right] &= \Pr\left[\overline{A_t} : \mathcal{F}_t\right](Y_t - 0) + (1 - \Pr\left[\overline{A_t} : \mathcal{F}_t\right])\,\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] \\
&= \Pr\left[\overline{A_t} : \mathcal{F}_t\right](Y_t - 0) \\
&\quad +(\epsilon - \Pr\left[\overline{A_t} : \mathcal{F}_t\right])\,\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] \\
&\quad +(1-\epsilon)\,\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] \\
&\leq \Pr\left[\overline{A_t} : \mathcal{F}_t\right] Y_0 + (\epsilon - \Pr\left[\overline{A_t} : \mathcal{F}_t\right])Y_0 + (1-\epsilon)\,\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] \\
&= \epsilon Y_0 + (1-\epsilon)\,\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right]. \qquad (11)
\end{aligned}$$

The inequality step uses the fact that $Y_0 \geq Y_t \geq 0$ for all t, so that Y_0 is an upper bound on any jump, and that $\epsilon - \Pr\left[\overline{A_t} : \mathcal{F}_t\right]$ is nonnegative, which is just a restatement of (4).

Now let us bound $\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right]$. Expanding the definitions (8) and (10) gives

$$\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] = \mathrm{E}\left[\int_{f(X_{t+1})}^{f(X_t)} \frac{1}{m_z} dz : \mathcal{F}_t, A_t\right]. \qquad (12)$$

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 (7). Inverting gives $\frac{1}{m_z} \leq \frac{1}{\mu_{f(X_t)}}$, and plugging this inequality into (12) gives

$$\begin{aligned}
\mathrm{E}\left[Y_t - Y_{t+1} : \mathcal{F}_t, A_t\right] &\leq \mathrm{E}\left[\int_{f(X_{t+1})}^{f(X_t)} \frac{1}{\mu_{f(X_t)}} dz : \mathcal{F}_t, A_t\right] \\
&= \mathrm{E}\left[\frac{1}{\mu_{f(X_t)}}\left(f\left(X_{t+1}\right) - f\left(X_t\right)\right) : \mathcal{F}_t, A_t\right] \\
&= \frac{1}{\mu_{f(X_t)}}\,\mathrm{E}\left[f(X_t) - f(X_{t+1}) : \mathcal{F}_t, A_t\right] \\
&\leq \frac{1}{\mu_{f(X_t)}}\mu_{f(X_t)} \\
&= 1. \qquad (13)
\end{aligned}$$

In the third step, we can pull the $\frac{1}{\mu_{f(X_t)}}$ term out of the expectation because $\mathcal{F}_t$ is generated by X_t (among other variables) and thus $\frac{1}{\mu_{f(X_t)}}$, a function of X_t, is measurable $\mathcal{F}_t$.

Applying (13) to (11) gives

$$\mathrm{E}[Y_t - Y_{t+1} : \mathcal{F}_t] \leq \epsilon Y_0 + (1-\epsilon). \qquad (14)$$

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 $Z_t = Y_t + \min(t, \tau)\left(\epsilon Y_0 + (1-\epsilon)\right)$. Conditioning on $t < \tau$, observe that

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

Alternatively, if $t \geq \tau$ we have $Y_{t+1} = Y_t = 0$, $\min(t, \tau) = \min(t+1, \tau) = \tau$, and thus $Z_{t+1} = Z_t = 0 + \tau(\epsilon Y_0 + (1-\epsilon))$. So in this case

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

In either case, $\mathrm{E}[Z_t - Z_{t+1} : \mathcal{F}_t] \leq 0$, implying $Z_t \leq \mathrm{E}[Z_{t+1} : \mathcal{F}_t]$. 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 $Z_0 = Y_0 \leq \mathrm{E}[Z_\tau] = \mathrm{E}[0 + \tau(\epsilon Y_0 + (1-\epsilon))] = (\epsilon Y_0 + (1-\epsilon))\,\mathrm{E}[\tau]$. Solving for $\mathrm{E}[\tau]$ then gives

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

∎

4.2 Lower bounds on greedy routing

We will now describe our lower bounds on the expected time for greedy routing on a line. As elsewhere, we assume that the graph consists of a sequence of nodes, each of which is connected to at most expected ℓ other nodes, including its immediate neighbors on either side. For values of ℓ up to $O(\log n)$, we consider two variants of the greedy routing algorithm: *one-sided routing*, where the search is not allowed to go past the target and reverse direction, and *two-sided routing*, where the search always chooses the nearest point to the target it can, regardless of which side of the target this point appears on. For one-sided routing, we show that $\Omega(\log^2 n/(\ell^2 \log\log n))$ expected steps are needed starting at a uniformly-chosen node, and for two-side routing, we show that $\Omega(\log^2 n/(\ell \log\log n))$ steps are needed. These results are stated and proved in Theorem 10. Details of the two routing variants are given in Section 4.2.1.

Lower bound for a superlogarithmic number of links For large values of ℓ, a lower bound of $\Omega(\frac{\lg n}{\lg \ell})$ on the worst-case routing time can be derived very simply, as shown in Theorem 3.

Theorem 3 *Let $\ell \in (\lg n, n^c]$. Then for any link distribution and any routing strategy, the delivery time $T = \Omega(\frac{\log n}{\log \ell})$.*

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, $\ell^T \geq n$. This gives a lower bound of $\Omega(\frac{\log n}{\log \ell})$ on T. ∎

Lower bound for a logarithmic number of links. For smaller numbers of links, the situation is more interesting. We consider the case of the expected outdegree of each node falling in the range $[1, \lg n]$. 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 links for other nodes.[2] Since we assume that each node is connected to its immediate neighbors, we require that ± 1 appears in Δ.

We consider 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

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

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 $\Omega(\log^2 n/(\ell \log\log n))$ 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 $\Omega(\log^2 n/(\ell^2 \log\log n))$, with some constraints on the distribution. We conjecture that these constraints are unnecessary, and that $\Omega(\log^2 n/(\ell \log\log n))$ is the correct lower bound for both models.

A formal statement of these results appears as Theorem 10 in Section 4.2.5, but before we can prove it we must develop machinery that will be useful in the proofs of both the one-sided and two-sided lower bounds. We now give a quick roadmap of the proof:

- Section 4.2.1 gives notation for link distributions.
- Section 4.2.2 defines the aggregate Markov chain obtained by considering the parallel trajectories of a range of initial points given the same link offsets at each step, describes how the states of this aggregate chain evolve as a function of the link distribution (Lemmas 4), shows that the aggregate chain correctly summarize the behavior of the original chain given a uniformly random starting point (Lemma 5), and gives a simple bound on how quickly the size of a range in the aggregate chain is likely to drop regardless of the link distribution (Lemma 6).
- Section 4.2.3 introduces the effects of the link distribution, and characterizes (Lemma 7) the number of *boundary points*—points in the range of initial locations at which the outgoing link chosen changes—that appear within a given range as function of the link distribution and the choice of one- or two-sided routing.
- Section 4.2.4 uses the boundary point definition of Section 4.2.3 to show that the speed at which the log of the size of the range in the aggregate chain drops is directly related to how many boundary points land in the central part of the range. Lemma 9, the main lemma in this section, characterizes formally the intuition that progress can only be made by links close to the size of the current range, and is used in the following section to bound the speed at which the log of the size of the range drops as a function of the link distribution and the current size.
- Finally, Section 4.2.5 contains the statement and proof of the main lower bound theorem, Theorem 10. The proof is a direct application of Theorem 2, using the machinery developed in the previous sections to show that the necessary technical conditions hold, and using a partitioning argument to bound the resulting integral.

A further Section 4.2.6, suggests how, with a more sophisticated proof, it might be possible to drop the extra conditions on the link distribution in Theorem 10.

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. This permits the algorithm always to make progress toward zero, assuming that it starts at a positive location, even if subtracting every other δ in Δ would carry it past zero. 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'.

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, and may consists of arbitrary contiguous intervals of points that all share the same sign. The aggregate states are characterized formally in Lemma 4, but before stating this lemma we still need a few definitions.

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 $\{x - \delta : x \in A\}$.

[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.

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

$$\Pr\left[S^{t+1} = S^t_{\Delta i\sigma} - \Delta_i : \Delta\right] = \frac{|S^t_{\Delta i\sigma}|}{|S^t|}, \tag{15}$$

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

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 $S^{t-1}_{\Delta i\sigma}$. ∎

Lemma 5 shows that moving to the aggregate chain does not misrepresent the underlying single-point chain:

Lemma 5 *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. ∎

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

$$\Pr\left[|S^{t+1}| \le a^{-1}|S^t| : S^t\right] \le 3\ell a^{-1}. \tag{16}$$

Proof:

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

Conditioning on Δ, there are at most $3|\Delta|$ non-empty sets $S^t_{\Delta i\sigma}$. If $|S^t_{\Delta i\sigma}| \le a^{-1}|S^t|$, then $|S^t_{\Delta i\sigma}|$ is chosen with probability at most a^{-1} by (15). 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

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

∎

Another way to write (16) is to say that $\Pr\left[\ln|S^t| - \ln|S^{t+1}| \ge \ln a : S^t\right] \le 3\ell a^{-1}$, which will give the bound (4) on the probability of large jumps when it comes time to apply Theorem 2.

4.2.3 Boundary points

Lemma 6 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 7 *Fix* S *and* Δ *and let* β *be defined as above. Suppose that* S *is positive. Let* $M = \{\min(S_{\Delta i\sigma}) : S_{\Delta i\sigma} \ne \emptyset\}$ *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 6 and Lemma 7 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 9.

Lemma 8 *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*

$$\frac{\sum_{i\in B} x_i \ln x_i}{\sum_{i\in B} x_i} \geq \ln\left(\max\left(c, \frac{M}{n}\right)\right). \tag{17}$$

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 (17) 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 9 *Fix $a > 1$, and let $S = S^t$ be a positive range with $|S| \geq a$. Define β as in Lemma 7. Let $S' = [\min(S) + \lceil a^{-1}|S|\rceil - 1, \max(S) - 1]$. Let A be the event $\left[\ln|S^t| - \ln|S^{t+1}| < \ln a\right]$. Then*

$$\mathrm{E}\left[\ln|S^t| - \ln|S^{t+1}| : S^t, A\right] \leq \ln\frac{1}{1-a^{-1}} + \frac{\ln \mathrm{E}[1+Z : S^t]}{\Pr[A : S^t]}, \tag{18}$$

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; the intent is that the large ranges are precisely those that yield $\ln|S^t| - \ln|S^{t+1}| < \ln a$. Observe that for any large $S_{\Delta i\sigma}$, $|S_{\Delta i\sigma}| > a^{-1}|S| \geq 1$, 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 7, we can characterize the elements of T as follows.

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

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

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

where the first inequality follows from Lemma 8. Now let us compute

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

In the second-to-last step, we use $\mathrm{E}[\ln |T| : S^t, A] \leq \mathrm{E}[\ln |T| : S^t] / \Pr[A : S^t]$, which follows from $\mathrm{E}[\ln |T| : S^t] = \mathrm{E}[\ln |T| : S^t, A] \Pr[A : S^t] + \mathrm{E}[\ln |T| : S^t, \neg A] \Pr[\neg A : S^t]$. In the last step, we use $\mathrm{E}[\ln |T| : S^t, A] \leq \ln E[|T| : S^t, A]$, 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 10 *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 link 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*

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

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

$$\Omega\left(\frac{\log^2 n}{\ell^2 \log\log n}\right), \tag{20}$$

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 $S^0 = \{1 \dots n\}$.

We are going to apply Theorem 2 to the sequence $S^0, S^1, S^2, \dots$ in the aggregate chain defined in Section 4.2.2, while defining $f(S) = \ln|S|$. We have chosen f so that when we reach the target, $f(S) = 0$ (by Lemma 4: when we reach the target, $S = \{0\}$). We assume that the target is reached at a finite time with probability 1, so that $f(S^t) = \ln|S^t|$ satisfies (2); if this is not the case, then the lower bound on the expected time to reach the target is ∞. It is also immediate from the definition of the chain that $|S^t|$ never increases, so that $f(S^t)$ satisfies (3). Finally, we know from Lemma 5 that the behavior of the aggregate chain models the behavior of the original routing algorithm given a uniform starting position. Thus, 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 (8) is large.

Let $a = 3\ell \ln^3 n$. By Lemma 6, for all t, $\Pr\left[|S^{t+1}| \le a^{-1}|S^t| : S^t\right] \le 3\ell a^{-1} = \ln^{-3} n$, and thus $\Pr[\ln|S^t| - \ln|S^{t+1}| \ge \ln a : S^t] \le \ln^{-3} n$. This satisfies (4) 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 $\{1 \dots |S^t|\}$ 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 that we use the monotonicity assumptions on the distribution Δ. The essential idea is that $f(S)$ drops no faster when S is of the form $\{x+1 \dots x+k\}$ than it does when S is $\{1 \dots k\}$, because moving S further away from the origin decreases its probability of being split. A complication is that it is not enough to look at where the elements of Δ land, as we do with one-sided routing, because (as described in Section 4.2.3), the sets $S_{\Delta i\sigma}$ may also split across boundary points that are midpoints (appropriately rounded) of adjacent elements of Δ. The additional constraints on the Δ distribution are used below to get an upper bound on the number of such boundary points in S that is non-decreasing S shifts away from zero. The upper bound is rather crude, as it counts the midpoints of every pair of points in Δ, whether or not they are adjacent. This crude estimate accounts for the squaring of ℓ in the two-sided lower bound, and it may be that a more sophisticated argument could eliminate this difference between the one-sided and two-sided lower bounds.

We now show the promised bound, nonincreasing with distance from the origin, on the probability that a given point is a member of the set of boundary points β. Suppose that the conditions on Δ for two-sided routing hold, i.e., 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 $\hat{\beta} = \left\{\text{absceil}\left(\frac{x+y}{2}\right) : x, y \in \Delta, x \neq y\right\}$, where $\text{absceil}(z)$, the *absolute ceiling* of z, is $\lceil z \rceil$ when $z \geq 0$ and $\lfloor z \rfloor$ when $z \leq 0$. Observe that $\hat{\beta} \supseteq \beta$, because β consists precisely of those elements of $\hat{\beta}$ for which there is no element of Δ between x and y. We will now show that the probability q_δ that δ appears in $\hat{\beta}$ is non-decreasing in $|\delta|$.

Rather than deal with the division by 2 and rounding immediately, we instead begin by counting the number of pairs $x, y \in \Delta$ that sum to each given value z. For each z, the expected number of distinct pairs x, y with $x + y = z$ and $x, y \in \Delta$ is at most $b_z = \sum_{i=-\infty}^{\infty} p_{z-i} p_i$. The sequence b_z is thus 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 $0 \leq z < z'$, $b_z \geq b_{z'}$, and similarly $b_{-z} \geq b_{-z'}$.

To go from b to q, we must take into account the mapping of $x+y$ to $\text{absceil}\left(\frac{x+y}{2}\right)$. Observe that for each $\delta \neq 0$, $\text{absceil}\left(\frac{x+y}{2}\right) = \delta$ if and only if $x+y$ is either $2\delta - \operatorname{sgn}\delta$ or 2δ. So $q_\delta = b_{2\delta - \operatorname{sgn}\delta} + b_{2\delta}$ 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 $\Pr[\delta \in \beta]$. From symmetry and unimodularity of b, we have that $q_\delta \geq q_{\delta'}$ and $q_{-\delta} \geq q_{-\delta'}$ whenever $0 < \delta < \delta'$. This gives us our desired monotonicity property for q.

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 $|S^t|$, by summing up p_i and q_i over the range 1 to $|S^t|$. This allows us to proceed in essentially the same way as in the one-sided case, considering only $|S^t|$ without regard to the actual position of S^t.

For convenience, formally define $p_i = \Pr[i \in \Delta]$ and $q_i = 0$ for one-sided routing. This permits writing a single argument in terms of the p_i and q_i that covers both the one-sided and two-sided cases, which we now proceed to do.

We will simplify some of the summations by first summing the p_i and q_i over certain predefined 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 at most $2\ell + \ell^2$ for two-sided routing.

Consider some $S = S^t$. Let A be the event $\left[\ln|S^t| - \ln|S^{t+1}| < \ln a\right]$. If $|S| \geq a$, then by Lemma 9 we have

$$\mathrm{E}\left[\ln|S^t| - \ln|S^{t+1}| : S^t, A\right] \leq \ln\frac{1}{1-a^{-1}} + \frac{\ln \mathrm{E}\left[1 + Z : S^t\right]}{\Pr[A : S^t]}, \tag{21}$$

where $Z = 2|\Delta \cap S'|$ with one-sided routing and $Z = 2|\Delta \cap S'| + |\beta \cap S'|$ with two-sided routing, with $S' = [\min(S) + \left\lceil a^{-1}|S|\right\rceil - 1, \max(S) - 1]$ in each case, as in Lemma 9.

As we observed earlier, our choice of a and Lemma 6 imply $\Pr[\ln|S^t| - \ln|S^{t+1}| \geq \ln a : S^t] \leq \ln^{-3} n$, so $\Pr[A : S^t] = 1 - \Pr[\ln|S^t| - \ln|S^{t+1}| \geq \ln a : S^t] \geq 1 - \ln^{-3} n \geq \frac{1}{2}$ for sufficiently large n. So we can replace (21) with

$$\mathrm{E}\left[\ln|S^t| - \ln|S^{t+1}| : S^t, A\right] \leq \ln\frac{1}{1-a^{-1}} + 2\ln \mathrm{E}\left[1 + Z : S^t\right], \tag{22}$$

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

$$\ln \mathrm{E}[1+Z] \leq \ln \left(1 + \sum_{i=\lceil a^{-1}|S| \rceil - 1}^{|S|-1} 2p_i + q_i \right), \tag{23}$$

and thus $\mathrm{E}\left[\ln |S^t| - \ln |S^{t+1}| : S^t, A\right]$ is bounded by

$$\mu_{\ln |S|} = \ln \frac{1}{1-a^{-1}} + 2 \ln \left(1 + \sum_{i=\lceil a^{-1}|S| \rceil - 1}^{|S|-1} 2p_i + q_i \right), \tag{24}$$

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

Let us now compute m_z, as defined in (7). 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 (24) both lie between $\left\lceil a^{-1}e^z \right\rceil - 1$ and $ae^z - 1$, so that

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

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

Finally, compute

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

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 $\frac{1}{c + 2\ln(1+x)}$ is convex and decreasing, we have

$$\begin{aligned} T(\ln n) &\geq \sum_{i=0}^{\lfloor \ln n / \ln a \rfloor - 1} \frac{\ln a}{\ln \frac{1}{1-a^{-1}} + 2 \ln(1 + \gamma_i + \gamma_{i+1} + \gamma_{i+2})} \\ &\geq \sum_{i=0}^{\lfloor \ln n / \ln a \rfloor - 1} \frac{\ln a}{\ln \frac{1}{1-a^{-1}} + 2 \ln \left(1 + \frac{L}{\lfloor \ln n / \ln a \rfloor}\right)} \\ &= \frac{\ln a \, \lfloor \ln n / \ln a \rfloor}{\ln \frac{1}{1-a^{-1}} + 2 \ln \left(1 + \frac{L}{\lfloor \ln n / \ln a \rfloor}\right)}. \end{aligned} \tag{25}$$

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 $\ln(1+x) \le x$ for $x \ge 0$. Thus

$$\begin{aligned} 2 \ln \left(1 + \frac{L}{\lfloor \ln n / \ln a \rfloor}\right) &\le 2 \frac{L}{\lfloor \ln n / \ln a \rfloor} \\ &= O\left(\frac{L(\log l + \log \log n)}{\log n}\right), \end{aligned}$$

and the bound in (25) simplifies to $\Omega\left(\log^2 n / \left(L(\log \ell + \log \log n)\right)\right)$. We can further assume that $\ell = O(\log^2 n)$, since otherwise the bound degenerates to $\Omega(1)$, and rewrite it simply as $\Omega\left(\log^2 n / \left(L \log \log n\right)\right)$.

For large L, the approximation $\ln(1+x) \le 1 + \ln x$ for $x \ge 0.59$ is more useful. In this case (25) 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 and gives essentially the same bound as Theorem 3.

We are not quite done with Theorem 2 yet, as we still need to plug our T and ϵ into (9) to get a lower bound on $\mathrm{E}[\tau]$. But here we can simply observe that $\epsilon T = O(1/\log n)$, so the denominator in (9) goes rapidly to 1, so that (9) simplifies to

$$T(\ln n) = \Omega\left(\frac{\log^2 n}{L \log \log n}\right). \tag{26}$$

Our stated bounds are thus finally obtained by substituting $O(\ell)$ or $O(\ell^2)$ for L, depending on whether we assume one-sided or two-sided routing. ∎

4.2.6 Possible strengthening of the lower bound

Examining the proof of Theorem 10, both the ℓ^2 that appears in the bound (20) 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 11 *Let G, Δ, and ℓ be as in Theorem 10. 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

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

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 10.

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 to 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 the cases of a single long-distance link and of multiple ones, both in a failure-free network and in a network 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.

4.3.1 Single long-distance link

We first analyze the delivery time in an idealized model with no failures and with one long-distance link per node. Kleinberg [10] 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 $d = 1$ and $l = 1$ to give a $O(\log^2 n)$ delivery time, we get a much simpler proof by use of Lemma 1. We include the proof below for completeness.

Theorem 12 *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 $T(n) = O(\log^2 n)$.*

Proof: Let the destination node be t, and let μ_k be the expected number of nodes crossed when the message is at a node s, at distance k from t. Clearly, μ_k is non-decreasing. To calculate a lower bound for it, we consider the following possibilities for the one long-distance link of s:

1. The link points to a node closer to t than s is without overshooting t.
2. The link points to a node in the direction opposite to t's. Let that node be at distance d_1 from t.
3. The link overshoots t by distance at most k.
4. The link overshoots t by distance greater than k, say d_2.

Since we are using two-sided routing, the long-distance link is taken only in cases (1) and (3), while in (2) and (4) the link of length 1 to the immediate neighbor in the direction of t is used. We can now express μ_k as follows.

$$\mu_k = \frac{\sum_{i=1}^{k} \frac{1}{i} \cdot i}{S} + \frac{\sum_{i=1}^{d_1-k} \frac{1}{i} \cdot 1}{S} + \frac{\sum_{i=1}^{k-1} \frac{1}{2k-i} \cdot i}{S} + \frac{\sum_{i=2k}^{d_2+k} \frac{1}{i} \cdot 1}{S},$$

where

$$S = \sum_{i=1}^{d_1-k} \frac{1}{i} + \sum_{i=1}^{d_2+k} \frac{1}{i} = H_{d_1-k} + H_{d_2+k} < 2H_n.$$

Then

$$\mu_k > \frac{1}{S}[k + H_{d_1-k} + 0 + (H_{d_2+k} - H_{2k})] > \frac{k}{S} > \frac{k}{2H_n}.$$

Since μ_k is non-decreasing, we can use Lemma 1 to get

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

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

4.3.2 Multiple long-distance links

The next interesting question is whether we can improve the $O(\log^2 n)$ delivery time by using multiple long-distance links instead of a single one. In addition to improvement in performance, multiple links give provide 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 long-distance links from each node. Figure 4.3.2 shows an example of a node's long-distance links for ℓ=3. We partition the set of possible values for ℓ in two ranges $[1, \lg n]$ and $(\lg n, n^c]$ and consider different strategies for generating links and for routing for each range.

In [11], Kleinberg uses a group structure to get a delivery time of $O(\log n)$ for the case of a polylogarithmic number of links. 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.

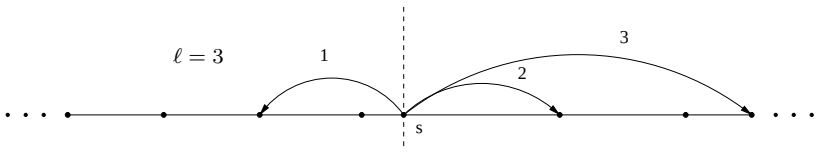


Figure 3: Multiple long-distance links for each node.

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

Theorem 13 *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 distances from the node. Let $\ell \in [1, \lg n]$. Then the expected delivery time is $T(n) = O(\log^2 n/\ell)$.*

Proof: The basic idea for this proof comes from Kleinberg's model [10]. 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 $O(\log n)$, thus giving a total upper bound of $O(\log^2 n)$ on the delivery time. We use the same phase structure in our model, and this proof is along 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 $q = \frac{d(u,v)^{-1}}{\sum_{u \neq v} d(u,v)^{-1}}$. We can get a lower bound on this probability as follows:

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

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

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

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

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

Now if X denotes the total number of steps, then $X = \sum_{j=0}^{\lg n} X_j$, and by linearity of expectation, we get $EX \leq (1 + \lg n)(8H_n/\ell) = O(\log^2 n/\ell)$. ∎

For $\ell \in (\lg n, n^c]$, we adopt a different linking model: the location of each node is represented as a number in a base $b \geq 2$, and links are generated to nodes at distances $1x, 2x, 3x, \ldots, (b-1)x$, for

each $x \in \{b^0, b^1, \ldots, b^{\lceil \log_b n \rceil - 1}\}$. The routing strategy is now deterministic and works 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 kind of routing is similar in spirit to Plaxton's algorithm [17]. The base b is a parameter of the model and can be used to control the tradeoff between number of links and search time, if the value of n is known or can be estimated.

Some special cases are instructive. Let $\ell = O(\log n)$ and let each node link to nodes in both directions at distances $2^i, 1 \leq i \leq 2^{\log n - 1}$, provided nodes are present at those distances. This gives $T(n) = O(\log n)$. Similarly let $\ell = O(\sqrt{n})$. Links are established in both directions to existing 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, $T(n) = O(1)$ when $b = n^c$, for any fixed c.

Theorem 14 *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 is $T(n) = O(\log_b n)$.*

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

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

Hence

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

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

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

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

4.3.3 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 $\ell \in [1, \lg n]$.

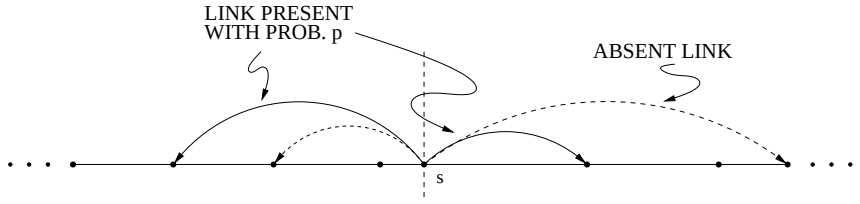


Figure 4: Each long-distance link is present with probability p.

Our proof is along similar lines as our proof for the case of 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 $O(\log n/p\ell)$, which gives a total delivery time of $O(\log^2 n/p\ell)$. 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. In some cases, this may not be a reasonable assumption; however, to remove it we must provide a mechanism that will allow the search to continue even if a node cannot find a live neighbor closer to the target than itself. Two possible mechanisms, namely backtracking and random re-route, are described in Section 6, where we also give experimental results of their application.

Theorem 15 *Let the model be as in Theorem 13. 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* $O(\log^2 n/p\ell)$.

Proof: 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 13. Let X_j be the number of steps spent in phase j. Then

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

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

We turn to the deterministic strategy with $\ell \in (\lg n, n^c]$ links. A similar intuition holds 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 delivery time increases since the message has to take shorter and shorter links.

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 $O(b \log n/p)$. As p decreases, the delivery time increases; whereas as b decreases, the delivery time also decreases, but the information stored at each node 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 distance $b^{\lfloor \log_b k \rfloor}$. 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

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

∎

4.3.4 Failure of nodes

We consider two different cases of node failures. In the first case, as described in Section 4.3.4.1, some of the nodes may fail and then the remaining nodes link to each other as per the link distribution. In the second case, analyzed 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 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 12. Let each node be present with probability p and all nodes link only to existing nodes. Then the worst-case expected delivery time is $O(\log^2 n)$.*

Proof: Recall the setting of the proof for Theorem 12. The methodology and definitions carry over to this proof without any change except for a slight adjustment in the computation of the expected drop μ_k to account for the probability of failure:

$$\begin{aligned}
\mu_k &= \frac{\sum_{i=1}^{k} \frac{1}{i} \cdot i \cdot p}{p \cdot S} + \frac{\sum_{i=1}^{d_1-k} \frac{1}{i} \cdot 1 \cdot p}{p \cdot S} + \frac{\sum_{i=1}^{k-1} \frac{1}{2k-i} \cdot i \cdot p}{p \cdot S} + \frac{\sum_{i=2k}^{d_2+k} \frac{1}{i} \cdot 1 \cdot p}{p \cdot S} \\
&> \frac{1}{S}[k + H_{d_1-k} + 0 + (H_{d_2+k} - H_{2k})] \\
&> \frac{k}{S} > \frac{k}{2H_n}.
\end{aligned}$$

As before, using Lemma 1, we get $T(n) \leq \sum_{k=1}^{n} 1/\mu_k = O(H_n^2)$. ∎

This result is identical to the one we derive for the case of no node failures in Section 4.3.1. Such agreement is not surprising: if nodes link only to other existing nodes, the only difference is that we get a smaller random graph, without any effect on the routing algorithm or the delivery time. Thus, we get the same upper bound even if we assume that the nodes do not populate all grid points on the line but instead are binomially distributed on them.

4.3.4.2 General Failures The analysis for node failures is not as simple as that for link failures because we no longer have the important property of independence that we have in the latter case. In the case of link failures, the nodes first choose their neighbors and then it is possible that some of these links fail; thus, 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 accessibility of a target node from any other node depends only on the presence of the link between the two nodes in question.

In case of node failures, this important independence property is no longer present. 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 a third 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 in both cases. This complicates the performance analysis because it is no 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.

Another source of dependence between overlay node failures might be that physical node failures generally occur in a correlated fashion. However, we assume that the overlay network is sparsely populated and that, through hashing, the location of resources in the physical network is not correlated with the placement of the same resources in the overlay network.

For our analysis, we consider jumps only to one phase lower 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 13 and let each node fail with probability p. Then the expected delivery time for successful queries is $O(\log^2 n/(1-p)\ell)$.*

Proof: Let T be the time taken to drop down from layer j to layer $j-1$. Let m 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 m live nodes in it, is given by

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

Now m can vary between 1 and N. (Note that m cannot be 0 because if there are no live nodes in the lower layer, the routing fails at this point.) We get

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

Not surprisingly, the expected waiting time in a layer is inversely proportional to the probability of being connected to a node in the lower layer and to the probability of such a node being alive.

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

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, so its occurrence will not affect the delivery time considerably. We have not yet analyzed this situation 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 $\Omega(1/d(v,u))$. We describe a heuristic approach to construct and maintain a random graph with such an invariant.

As explained below, we allow each node u that joins the network before another node v to connect to v when v joins the network. As soon as a node is connected to one neighbor in the network, it is considered to be added to the network as it can be reached using the search process.

Thus, if multiple nodes are added at the same time, they can still connect to each other as each node chooses multiple neighbors to link to. Since the choice of links 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 links according to the inverse power law distribution with exponent 1 and connects to them by running the search algorithm. If a desired sink u is not present, v connects to u's closest live 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: $\text{Prob}[L \leq c \cdot k^{-1}] = 1 - (1 - c \cdot k^{-1})^{k-1}$. A lower bound on the probability that the link (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 among the $k-1$ points added before v. Otherwise, the aforementioned probability is 0, which means that we need to amend our linking strategy to transfer probability mass from the case of u having arrived before v to the case of u having arrived after v. We describe next how to accomplish this task.

Let v be a new point. We give earlier points the opportunity to obtain outgoing links to v by having v (1) calculate the number of incoming links it "should" have from points added before it arrived, and (2) choose such points according to the inverse power-law distribution with exponent 1.[5] If ℓ is the number of outgoing links for each point, then ℓ will also be the expected number of incoming links that v has to estimate in step (1). We approximate the number of links ending at v by using a Poisson distribution with rate ℓ.

After step (2) is completed by v, each chosen point u responds to v's request by choosing one of its existing links to be replaced by a link to v. The choice of the link to replace can vary. We use a strategy that builds on the work of Sarshar *et al.*[19]. In that work, the authors use ideas of Zhang *et al.*[22] to build a graph where each node has a single long-distance link to a node at distance d with probability $1/d$. When a node with a long-distance link at distance d_1 encounters a new node at distance d_2, either due to its arrival or due to a data request, it replaces its existing link with probability $p_2/(p_1 + p_2)$, where $p_i = 1/d_i$, and links to the new node. We extend this idea to our case of multiple long-distance links. Consider a node u with ℓ neighbors at distances $d_1, d_2, \ldots, d_\ell$. When a new node v at distance $d_{\ell+1}$ requests an incoming link from u, u replaces one of its existing links with a link to v with probability $p_{\ell+1}/\sum_{j=1}^{\ell+1} p_j$. This is a trivial extension of the formula $p_2/(p_1 + p_2)$ of [19]. However, this probability must now be distributed among u's ℓ existing long-distance links since u needs to choose one of them to redirect to v. We choose to do that according to the inverse power-law distribution with exponent 1, that is, u chooses to replace its link to the node at distance d_i, $1 \leq i \leq \ell$, with probability $(p_i/\sum_{j=1}^{\ell} p_j)$. Hence, the probability that u decides to link to v and decides to replace its existing link to the node at distance d_i with a link to v is equal to $(p_i/\sum_{j=1}^{\ell} p_j) \cdot (p_{k+1}/\sum_{j=1}^{\ell+1} p_j)$. Notice that u may decide not to redirect any of its existing links to v with probability $1 - p_{\ell+1}/\sum_{j=1}^{\ell+1} p_j$. The intuition for using such replacement strategy comes from the invariant that we want to maintain dynamically as new nodes arrive: u has a link to a node i at distance d_i with probability inversely proportional to d_i; hence, conditioning

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

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

on u having ℓ long-distance links, the following equation must hold.

$$\begin{aligned}
\text{Prob}[u \text{ replaces link to } i \text{ with link to } v] &= \text{Prob}[u \text{ has a link to } i \text{ before } v \text{ arrives}] \\
&- \text{Prob}[u \text{ has a link to } i \text{ after } v \text{ arrives}] \\
&= \frac{p_i}{\sum_{j=1}^{\ell} p_j} - \frac{p_i}{\sum_{j=1}^{\ell+1} p_j} \\
&= \frac{p_i}{\sum_{j=1}^{\ell} p_j} \cdot \frac{p_{\ell+1}}{\sum_{j=1}^{\ell+1} p_j}.
\end{aligned}$$

The same heuristic can be used for regeneration of links when a node crashes. Every node periodically checks to see that all of its neighbors are alive. When a node detects that a neighbor is not responding, it replaces its link to that neighbor by a link to a new node. The probability distribution used to choose the new neighbor is as given in the formula above.

To analyze the performance of the heuristic in practice, we used it to construct a network of 2^{15} nodes with 15 links each, ten separate times. The nodes were mapped uniformly at random to a real line segment with each node at a grid point. The length of a link corresponds to the number of grid points between the end-points of the link.

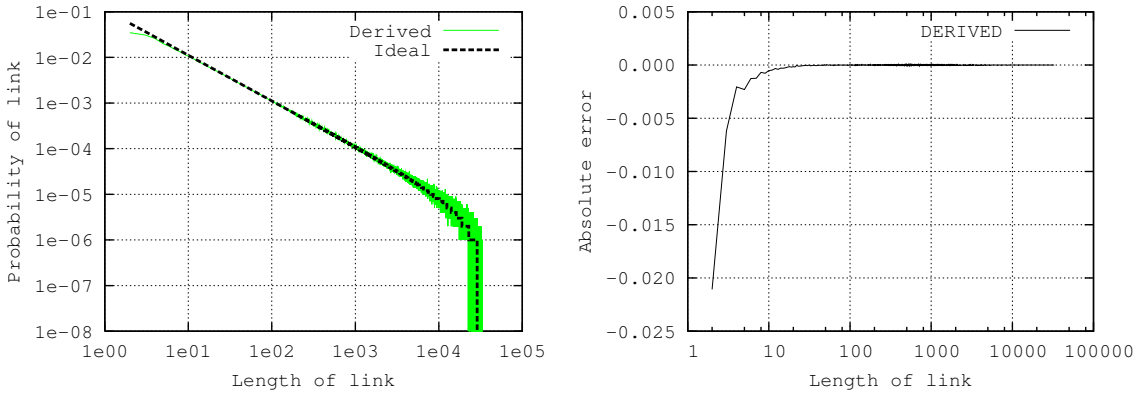
After averaging the results over the ten networks, we plotted the distribution of long-distance links derived from the heuristic, along with the ideal inverse power-law distribution with exponent 1, as shown in Figure 5(a). We see that the derived distribution tracks the ideal one very closely, with the largest absolute error being roughly equal to 0.022 for links of length 2, as shown in the graph of Figure 5(b). We also performed some experiments to compare the performance of the heuristic network to the ideal network with respect to the number of failed searches and the number of routing hops with node failures. We present those results in Section 6.

We also performed experiments for an alternative link replacement strategy: a node chooses its *oldest* link to replace with a link to the new node. The performance of this strategy is almost as good as the performance of our replacement strategy described previously. We omit those results because it is difficult to distinguish between the results of the two strategies on the scale used for our graphs.

There has also been other related work [16] 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 both ideal and heuristic networks of $n = 2^{15}$ nodes at the application level. In both the networks, each node is connected to its immediate neighbors and has $\lg n = 15$ long-distance links. In the ideal network, the links are chosen as per the inverse power-law distribution with exponent 1 as explained in Section 4.3. In the heuristic network, the links are chosen as per the heuristic described in detail in Section 5. 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 then repeatedly choose random source and destination nodes that have not failed and route a message between them. For each value of p, we ran 10 simulations, delivering



(a) The derived distribution.

(b) Absolute error.

Figure 5: (a) The distribution of long-distance links produced by the inverse-distance heuristic (DERIVED) compared to the ideal inverse power-law distribution with exponent 1 (IDEAL). (b) The absolute error between the derived distribution and the ideal inverse power-law distribution with exponent 1.

1000 messages in each simulation, and averaged the number of hops for successful searches and the 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.

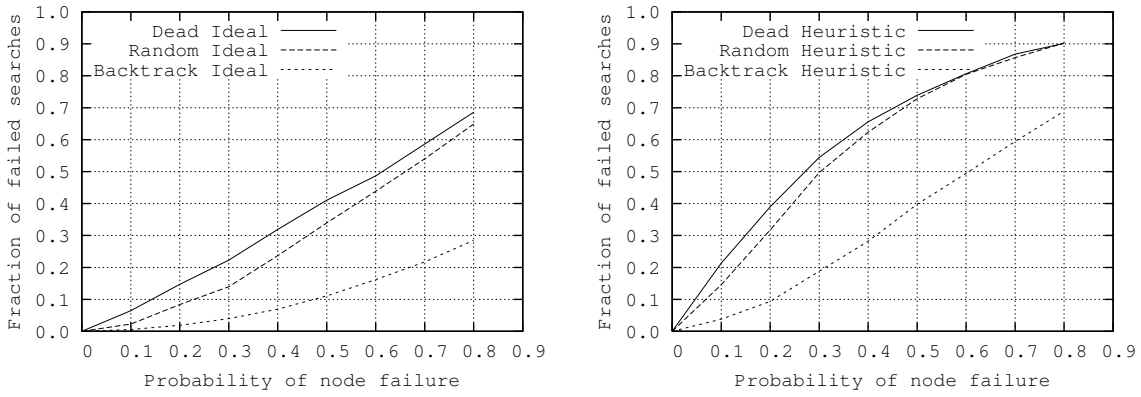
1. Terminate the search.

2. 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 explained in [21]).

3. 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 the message to.

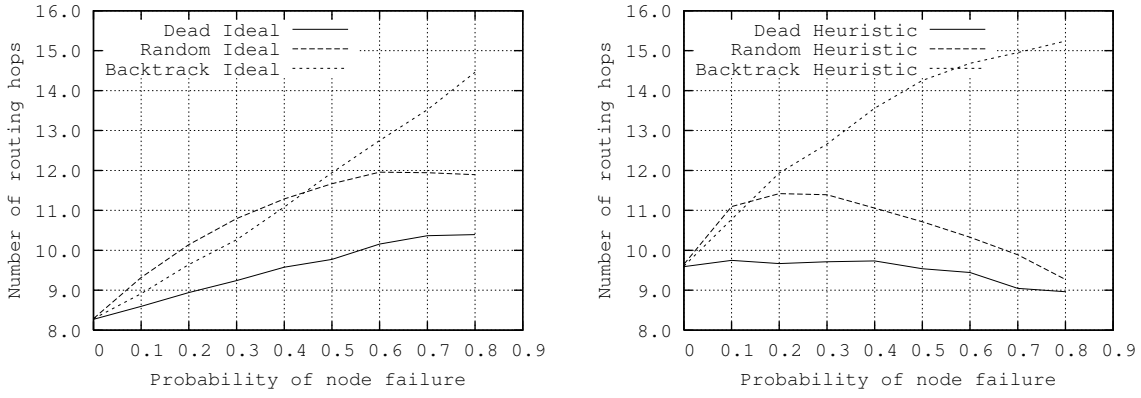
For all these strategies we note that once a node chooses its best neighbor, it does not send the message to any other neighbor if it finds out that the best neighbor has failed.

Figures 6(a) and 7(a) show the fraction of messages that fail to be delivered and the average number of routing hops versus the fraction of failed nodes in an ideal network. We see that this system behaves well even with 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. These results are very promising and it would be interesting to study backtracking analytically. We see that in the case of random rerouting, the average delivery time does not increase too much as the probability of node failure increases. This happens because quite a few of the searches fail, so the ones that succeed (with a few hops) lead to a small average delivery time.



(a) Ideal network. (b) Heuristic network.

Figure 6: The fraction of messages that fail to be delivered as a function of the fraction of failed nodes (a) in an ideal network and (b) in a heuristic network.



(a) Ideal network. (b) Heuristic network

Figure 7: The average number of routing hops for successful searches as a function of the fraction of failed nodes (a) in an ideal network and (b) in a heuristic network.

Figures 6(b) and 7(b) show the fraction of messages that fail to be delivered and the average number of routing hops versus the fraction of failed nodes in a heuristic network. We see that the heuristic network does not perform as well as the ideal network with respect to the number of failed searches, although we see similar trends like a significant improvement by using backtracking, and only a marginal improvement by using random rerouting instead of just terminating the search when it reaches a dead node. We believe that the heuristic network does not perform as well because many nodes may get isolated from the other nodes leading to a larger number of failed searches and shorter routing paths when the searches succeed. It would be interesting to study if using a different heuristic can give better performance in the network.

7 Conclusions and future work

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

Table 1: Summary of upper and lower bounds for routing.[6]

Table 1 summarizes our upper and lower bounds. 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 in a peer-to-peer system, 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 [9]). Furthermore, we simplified our analysis of node failures by assuming that physical node failures are not correlated with overlay node failures. Such correlation may seem unnatural, but since it is obscured by hashing we expect our results will not change significantly if the assumption is lifted. It is an interesting open problem to study such scenario analytically.

Other promising directions 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

[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.

Byzantine failures. We would also be very interested in a matching upper bound for the lower bound of Section 4.2 as well as in any other applications of the technical result of Theorem 2.

8 Acknowledgments

The authors are grateful to Ben Reichardt for pointing out an error in an earlier version of Lemma 9 and to the referees for their corrections and suggestions of interesting open problems for future work.

References

[1] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *Proceedings of the 15th International Symposium on Distributed Computing (DISC '01)*, pages 270–284, Linbon, Portugal, Oct. 2001.

[2] I. Benjamini and N. Berger. The diameter of long-range percolation clusters on finite cycles. *Random Structures Algorithms*, 19(2):102–111, 2001.

[3] I. Benjamini, H. Kesten, Y. Peres, and O. Schramm. The geometry of the uniform spanning forests: transitions in dimensions 4, 8, 12, ... *Annals of Mathematics*, 160:465–491, 2004.

[4] M. Biskup. On the scaling of the chemical distance in long-range percolation models. *Annals of Probability*, 32(4):2938–2977, 2004.

[5] D. Coppersmith, D. Gamarnik, and M. Sviridenko. The diameter of a long-range percolation graph. *Random Structures and Algorithms*, 21(1):1–13, 2002.

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

[7] F. Kaashoek and D. Karger. Koorde: a simple degree-optimal hash table. In *Proceedings of the 2nd International workshop on Peer-to-Peer Systems*, pages 98–107, Berkeley, CA, USA, Feb. 2003.

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

[9] 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, Crete, Greece, July 2001.

[10] J. Kleinberg. The small-world phenomenon:an algorithmic perspective. In *Proceedings of 32nd Annual ACM Symposium on Theory of Computing*, pages 163–170, Portland, Oregon, USA, May 2000.

[11] J. Kleinberg. Small-world phenomena and the dynamics of information. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[12] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing*, pages 183–192, Monterey, CA, USA, July 2002.

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

[14] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proceedings of the 15th Annual Symposium on Parallel Algorithms and Architectures*, pages 50–59, San Diego, California, June 2003.

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

[16] G. Panduragan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. *IEEE Journal on Selected Areas in Communications*, 21(6):995–1002, Aug. 2003.

[17] C. G. Plaxton, R. Rajaraman, 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)*, pages 311–320, Newport, RI, USA, June 1997.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–170, San Diego, CA, USA, Aug. 2001.

[19] N. Sarshar and V. Roychowdhury. A random structure for optimum cache size distributed hash table (DHT) peer-to-peer design. Oct. 2002. http://arxiv.org/abs/cs.NI/0210010.

[20] 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, San Diego, CA, USA, Aug. 2001.

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

[22] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve freenet performance. In *Proceedings of IEEE INFOCOM*, New York, NY, USA, June 2002.

[23] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), Jan. 2004.