Using the Small-World Model to Improve Freenet Performance

Hui Zhang* Ashish Goel†
University of Southern Californina

Ramesh Govindan ‡
International Computer Science Institute, Berkeley

***Abstract* – Efficient data retrieval in a peer-to-peer system like Freenet is a challenging problem. In this paper we study the impact of cache replacement policy on the performance of Freenet. We find that, with Freenet's LRU cache replacement, there is a steep reduction in the hit ratio with increasing load. Based on intuition from the small-world models and the recent theoretical results by Kleinberg, we propose an enhanced-clustering cache replacement scheme for use in place of LRU. Such a replacement scheme forces the routing tables to resemble neighbor relationships in a small-world acquaintance graph -- clustering with light randomness. In our simulation this new scheme improved the request hit ratio dramatically while keeping the small average hops per successful request comparable to LRU. A simple, highly idealized model of Freenet under clustering with light randomness proves that the expected message delivery time in Freenet is $O(\log^2 n)$ if the routing tables satisfy the small-world model and have the size $q(\log^2 n)$.**

I. INTRODUCTION

A peer–to–peer networked system is a collaborating group of Internet nodes which overlay their own special–purpose network on top of the Internet. Such a system performs application–level routing on top of IP routing. These systems share some of the same characteristics as the Internet in that they can grow to be quite large, may need to utilize distributed control and configuration, usually employ a naming scheme that allows them to address a node without knowing its exact whereabouts, and possess a routing mechanism that allows each node to meaningfully communicate with the rest of the system. Typically a peer–to–peer network performs a very specific function such as distributed data storage, cache replication, multicasting etc. and uses normal Internet functionality for all other purposes. These systems are diverse enough that it would not be desirable to make modifications to the IP routing/naming protocols to support each instance of such a system.

Peer–to–peer systems such as Napster [1] and Gnutella [2] have, of late, received a fair amount of attention in the non–technical literature. Perhaps proving that there is more to this technology than the hype surrounding it would suggest, several research groups have recently designed a variety of peer–to–peer systems: systems that provide infrastructure for flexibly evolving the Internet [3][4], and systems for large–scale network storage [5], anonymous publishing [6], and application–level multicast [7][8][9].

At the core of many of these systems [3][4][5][6] lie innovative and novel distributed algorithms for disseminating content. These systems can be modeled as distributed hash-tables; they essentially distribute ⟨key, data⟩ tuples across various nodes in a large network in a manner that facilitates scalable access to these tuples using the key. We taxonomize these systems into two categories: structured systems where the assignment of a key to the node on which its corresponding data is stored is determined by the structure of the key space, and unstructured systems where no such assignment exists. Systems like CAN [3], CHORD [4], and OceanStore [5] are examples of the former, and Freenet [6] is an instance of the latter.

Such systems can have interesting and unanticipated properties. In this paper we study the performance of an unstructured system, namely Freenet. We find that the hit ratio (the likelihood of finding the datum associated with a key within a fixed number of network hops) in Freenet is crucially dependent on the local policy used to manage the cache of data (called *datastore* in Freenet) and the routing table. A standard LRU–like cache replacement policy can result in significantly low hit ratio under high load (i.e. when the number of files stored in the network is high). This is an important finding. While memory is cheap these days, Freenet-like peer-to-peer systems will always have limits on cache sizes. Thus, examining the performance degradation of such systems under high load becomes important, especially, for example, in the context of understanding the immunity of these systems to denial-of-service attacks.

We observe that this particular performance degradation in Freenet results from the fact that LRU cache replacement results in routing tables that are not highly clustered. This is undesirable since the success and efficiency of the Freenet

* Deptartment of Computer Science, University of Southern California, huiz@pollux.usc.edu. Part of this work was completed when the author was at Information Science Institute, University of Southern California. Research supported in part by DARPA under contract number F30602-00-2-055.

† Deptartment of Computer Science and Information Science Institute, University of Southern California, agoel@pollux.usc.edu. Research supported in part by DARPA under contract number F30602-00-2-055.

‡ International Computer Science Institute, Berkeley, ramesh@ICSI.Berkeley.EDU. Part of this work was completed when the author was at Information Science Institute, University of Southern California. Research supported in part by DARPA under contract number F30602-00-2-055.

routing algorithm relies on a high degree of clustering of the routing table entries. Our solution to this performance degradation relies on two observations. First, we observe that the routing tables can be shaped by changing the cache replacement policy at each node; this need not include any changes to the Freenet routing protocol and is therefore incrementally deployable. Second, we use intuition from the small world model [10][11][16] which says that the routing distance in a graph is small if each node has pointers to its geographical neighbors (causing clustering) as well as some randomly chosen far away nodes. This intuition leads us to propose a simple clustered cache replacement scheme with a small amount of randomization.

We then perform a simulation study to compare the two schemes under high loads. Our proposed cache replacement scheme results in a significantly higher hit ratio compared to LRU, and also a significantly small number of average hops per request. The proposed scheme and LRU result in similar values for the average number of hops per successful request. It is interesting that such a small change in the system can lead to a significant improvement.

We also develop an idealized model of Freenet under our cache replacement scheme. We prove that the expected message delivery time in this idealized model is $O(\log^2 n)$, provided we have $O(\log^2 n)$ amount of memory per node. Here n is the number of nodes in the system. More structured systems such as CAN or CHORD [3][4]achieve a delivery time of O(log n) with O(log n) memory. It is surprising that an unstructured system such as Freenet can come so close to the structured systems, even in a highly idealized model. We believe that a theoretical understanding of unstructured systems in more realistic models is an important open problem.

The paper is organized as follows: In Section II we give a brief description of Freenet and the small-world model. In Section III we describe the preliminary simulation results for the original Freenet protocol. In section IV a simple enhanced-clustering cache replacement scheme is proposed and the simulation results show that the enhanced-clustering caching with random shortcuts outperformed both LRU caching and the enhanced-clustering caching without random shortcuts. Section V presents our analytical results and section VI concludes the paper.

II. FREENET AND THE SMALL-WORLD MODEL

In this section we briefly describe Freenet and the well-known small-world model.

A. *Freenet*

Freenet [6] is a distributed anonymous information storage & retrieval system. In Freenet, files are identified by binary file keys obtained by applying a hash function to a string that describes the contents of the file. For this reason, we use the words *key*, *file*, and *data* interchangeably in this paper. Each node maintains a routing table which is a set of ⟨ *key*, *pointer* ⟩ pairs, where *pointer* points to a node that has a copy of the file associated with *key*. A steepest-ascent hill-climbing search with backtracking is used to locate a document. Loop detection and a HopsToLive (Freenet's TTL) counter are added to this basic scheme to avoid request looping and exhaustive searching. Fig. 1 shows a typical sequence of request messages. A request for key 8 is initiated at node A. Node A forwards the request to node B, which forwards it to node C since in B's routing table C is the node who has the cloestest key to key 8. Node C is unable to contact any other nodes and returns a backtracking ``request failed" message to B. Node B then tries its second choice, D. Node D finds key 8 in its routing table and forwards the request to the corresponding node - E. The data is returned from E via D and B back to A, which ends this request sequence. The data is cached on D, B and A. An entry for key 8 is also created in the routing tables of D, B and A. Data inserts follow a similar strategy to requests [6].

In this alogrithm, there is no global consensus on where a document should be stored. Freenet chose an unstructured key space architecture due to the robustness and security considerations. The main premise behind Freenet is that, gradually the key space becomes more and more structured automatically due to the data request mechanism and the routing tables converge to a state where most of the queries are answered successfully and quickly. Freenet pays a lot of attention to anonymity and deniability, and replication is an integral part of the architecture.

In addition to the routing table, each Freenet node has a datastore. When a file is ultimately returned (forwarded) for a successful retrieval (insertion), the node passes the data to the upstream (downstream) requester, caches the file in its own datastore, and creates a new entry in its routing table associating the actual data source with the requested key. When a new file arrives (from either a new insert or a successful request) which would cause the datastore to exceed the designated size, the Least Recently Used (LRU) files are evicted in order until there is room. Routing table entries are also eventually replaced using an LRU policy as the table fills up. Note that although datastore cache replacement and route replacement are logically separate mechanisms, they both decide the content in the routing table. Cache replacement scheme decides which ⟨ key, pointer ⟩ pairs are put into the routing table by choosing the files to be cached and then generating the corresponding ⟨ key, pointer ⟩ pairs. Route replacement policy decides which ⟨ key, pointer ⟩ pairs are chosen to be deleted from the routing table when the routing table fills up. The size of the routing table is chosen with the intention that the entry for a file will be retained longer than the file itself.

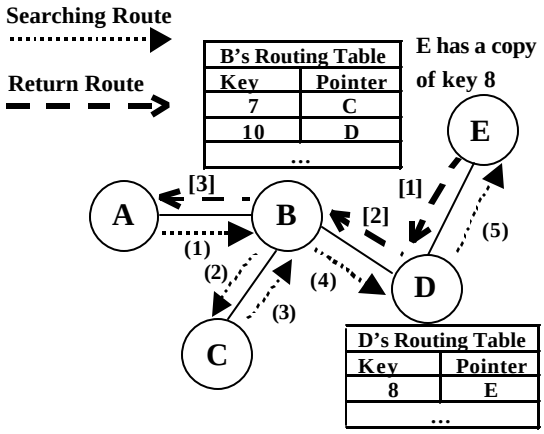


Fig.1 An example of a search in a Freenet network. Node A searches for Key 8 and finally finds it in E.

B. Small-world model

The small-world phenomenon is pervasive in networks arising from society, nature and technology. In many such networks, empirical observations suggest that any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances [10][12]. One network construction that gives rise to small-world behavior is where each node in the network knows its physical neighbors, as well as a small number of randomly chosen distant nodes. The latter represent shortcuts in the network. It has been shown that this construction leads to graphs with small diameter, leading to a small routing distance between any two individuals [10][16]. Kleinberg [12] defined an infinite family of network models that naturally generalized the model in [16] and then proved that there is exactly one model within this family for which a decentralized algorithm exists to find short paths with high probability. In this model, the probability of a random shortcut being a distance x away from the source is proportional to $1/x$ in one dimension, proportional to $1/x^2$ in two dimensions, and so on.

C. Freenet and the small-world phenomenon

An earlier study argued that a Freenet network evolve into a network with small-world characteristics [13]. In particular, the study showed that the median path length in a Freenet network scales logarithmically with the size of the network and its clustering coefficient is high, both defining characteristics of small-world networks. We did a similar simulation like that in [13] with our own simulator and validated the logarithmical relation between the average hops per request and the network size in Freenet under low load.

However, those simulations were conducted under low load. In their 1000-node simulation, the average number of files inserted into the network by one node is only 2.5. Under heavy load, more frequent local caching actions could break up clusters caused by the Freenet routing mechanism and cause these networks to evolve in a fashion that might not satisfy the small-world hypothesis.

We explore the behavior of Freenet under high load in the next section. Then, we use intuition from the small-world model to design a cache replacement scheme that attempts to preserve Freenet's small-world property even under high load.

III. SIMULATING FREENET PERFORMANCE UNDER HEAVY LOAD

Freenet's design focus on anonymity makes it hard to measure the global network directly, a fact already observed by the designers of Freenet [13]. For this reason, we resort to simulation to study the performance of Freenet. We implemented a stand-alone simulator† that mimics document generation, storage, routing, and retrieval in Freenet. In this section, we state the assumptions made in our simulation, define the metrics that we use in our evaluation of Freenet and present our simulation results.

Much of this paper is devoted to the study of Freenet under high "load". Our measure of load is the average number of files inserted by a node into the system, since we are interested in investigating the impact of cache replacement strategies.

A. Simulation Assumptions

Lacking data about actual file insertion workloads, we assume that all nodes generate data files and send requests at the same rate. Furthermore, we assume that all nodes have the same size of datastore and routing table, and that all data files are of the same size. While thses assumptions are somewhat unrealistic, they enable us to understand the impact of cache limits more easily. We believe that a more heterogeneous workload, or a more heterogeneous node capacity will not qualitatively affect our results.

In our simulations, we do not model the impact of node failures. Node failures are somewhat orthogonal to the impact of cache limits.

B. Performance Metrics

Intuitively, the performance of a routing system such as Freenet is well captured by the average path length required to access a data item. In practice, Freenet imposes a HopsToLive on data accesses. Therefore, the performance of the system is better represented by two metrics: the *request hit ratio* and the *average hops per request*.

The former is defined as the ratio of the number of successful requests to the total number of requests made. For reasons that will become later, we define two variants of the latter metric. The *average hops per request* is the ratio of the total number of hops incurred across all requests to the number of all requests. When a request fails, it is deemed to have incurred HopsToLive number of hops. The *average hops per successful request* is defined similarly, but only for requests that are successful.

† The simulator can be downloaded from http://netweb.usc.edu/~huizhang/freenet.html

C. *Freenet Performance under Varying Loads*

In this section, we illustrate the performance of Freenet under heavy load using a simple simulation. The duration of the simulation was 12,000 time steps, and the network had 300 nodes. Each node had a datastore limit of 40 files, a routing table limit of 90 files. The initial topology of the system is a ring: each node has pointer to two neighbors. This initial topolgy is imposed by Freenet routing tables, and need not have any relation to the underlying physical Internet topology. Each request is limited to 40 hops. Each node randomly generates and inserts a key (i.e., a file) with probability K per time step in the first 200 time steps (K varies in the range [0.005, 0.13]). All insertions are stopped after time 200. Each node generates a request for a random key with probability R=0.002 per time step throughout the whole simulation.

The simulation result in Fig. 2 shows that request hit ratio decreases significantly with an increase in the load in the network. To check whether this behavior was primarily due to cache limits, we repeated the above simulation but increased the datastore size to 200 and routing table size to 250. The simulation result in Fig. 3 shows a curve with the same shape.

In an attempt to understand this steep degradation, we plot the typical key distribution (at the end of the simulation) in Freenet node's datastore under light and heavy loads (Fig. 4 & Fig. 5, in which *retrieved times* is the times that the corresponding file is retrieved from this node due to the data requests from the other nodes. A value of 0 means this file is cached by the node but never requested by other nodes). Fig. 4 shows that under light load, local clustering is obvious. But this clustering characteristic vanishes with an increase in the number of files (keys) generated by this node (as shown in Fig. 5); the few big clusters under light load become lots of small clusters.

This disappearance of high local clustering appears to be responsible for the significantly low hit ratio. The HopsToLive mechanism prevents exhaustive search and Freenet nodes depend on local clustering to provide the direction to key-searching. If this were indeed true, we would expect that a cache management strategy that preserves key clustering under high load would exhibit a much more graceful system degradation. We investigate such a strategy in the next section.

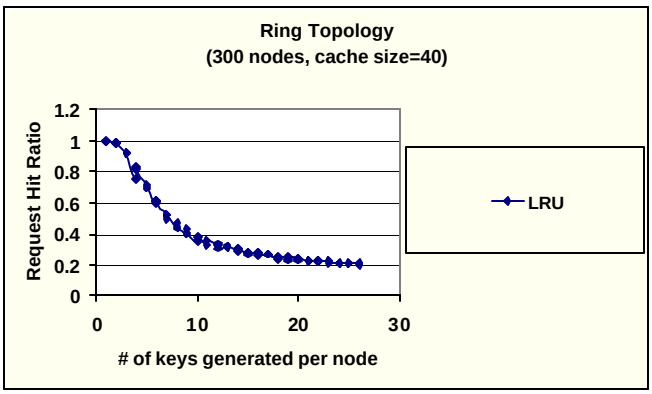


Fig. 2. The curve of hit ratio Vs. Load for Freenet with LRU scheme

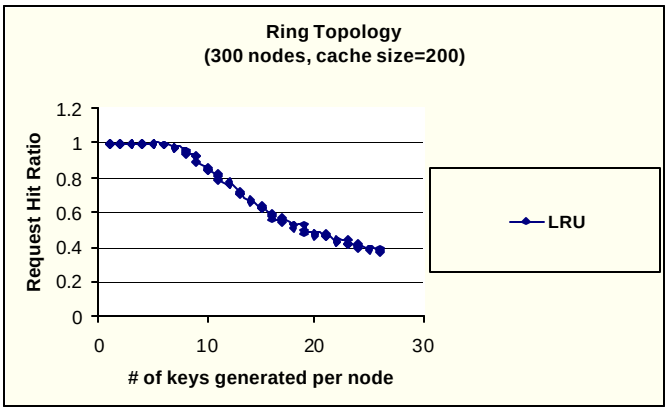


Fig.3. Simulation with a larger datastore and routing table. We see the hit ratio still decreased rapidly with the increasing of the load.

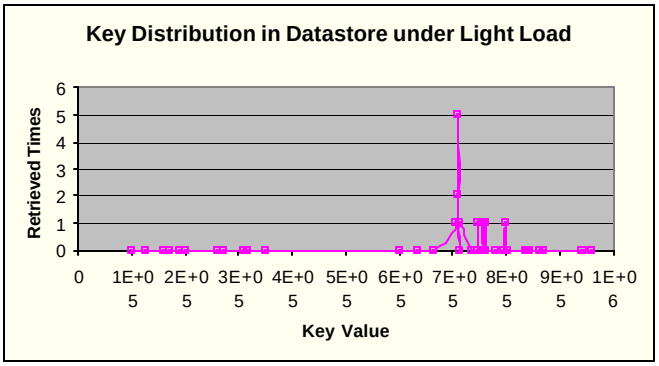


Fig.4. The files stored in the datastore cluster around the two keys generated by the node itself. Local clustering is obvious under light load (in this case average number of keys generated per node =2)

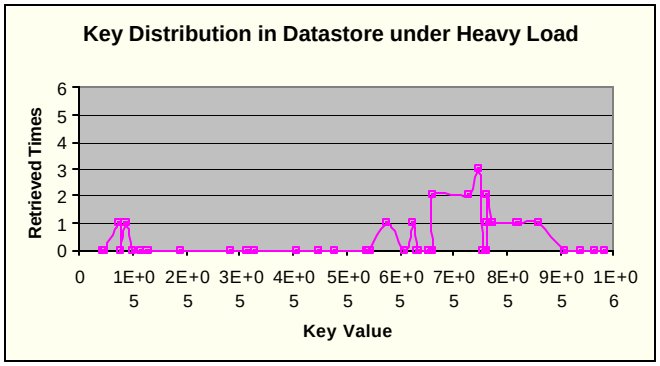


Fig.5. Due to the large number of keys generated by the node itself, the local clustering phenomenon becomes weak under heavy load (in this case average number of keys generated per node =20)

IV. ENHANCED-CLUSTERING CACHE REPLACEMENT SCHEME

We are now left with an interesting problem: how can we improve the routing performance of the Freenet protocol efficiently without affecting its design goals? Increasing the cache size or HopsToLive value may not always be possible or desirable. The cache-size is locally administered since it depends on individual system resources. Increasing the

HopsToLive value could increase the hit ratio, at the expense of significantly increased access latency. From Fig.6, the crucial observation is that we can achieve our performance goals by preserving key clustering in the cache and this can be done passively by changing the cache replacement policy without changing the Freenet routing protocol or sacrificing anonymity and deniability. Recall that when a datastore is full at a node, the node discards the least recently used files and creates a new ⟨ key, pointer⟩ tuple in the routing table for the new file to be cached. In order to shape the routing table, we use the following cache replacement scheme instead of LRU (note that LRU is still used for routing table replacement).

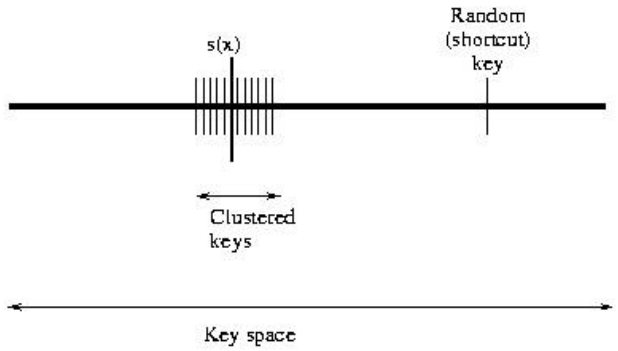


Fig.6. For the routing table at node x to conform to the small-world model, we need a set of key entries clustered around some key s(x) and one or more randomly chosen shortcut keys

A. *Enhanced-clustering Cache Replacement Scheme*

The enhanced-clustering cache replacement scheme consists of two parts:

1. Each node x chooses a seed s(x) randomly from the key space ***S*** when it joins the system.
2. When the datastore at a node is full and a new file with key u arrives (from either a new insertion or a successful request), the node finds out in the current datastore the file with key v farthest from the seed in terms of the distance in the key space ***S***.

   Distance (v, seed) = $Max_{w \in Datastore}$ Distance (w, s(x))

   (a) If Distance(u, seed) ≤ Distance (v, seed), cache u and evict v. Create an entry for u in the routing table. This has the effect of clustering the keys in the routing table around the seed of the node.

   (b) If Distance(u, seed) > Distance (v, seed), cache u, evict v and create an entry for u in the routing table with a probability p (randomness). This has the effect of creating a few random shortcuts.

B. *Comparison of Three Cache Replacement Schemes*

We compare three cache replacement schemes: *LRU*, *enforced-clustering*, and *enforced-clustering with random shortcuts*. LRU always throws out the least recently used file from the datastore. The only difference between *enforced-clustering* and *enforced-clustering with random shortcuts* is the value of the randomness p. Enforced-clustering implements the scheme outlined above with p=0 and therefore always throws out the key furthest from the seed of the node. It actually shapes the routing table to make the network conform to the regular-graph model. Enforced-clustering with random shortcuts implements the scheme outlined above with p = 0.03 and therefore still keeps small number of random shortcuts in the routing table. Intuitively, this scheme should make Freenet look more like a small-world network. The probability p=0.03 was chosen in order to achieve the highest hit ratio and the fewest Average hops per request for the range $0 \leq p \leq 1$ in simulation. It remains an interesting open problem to determine the best value of p as a function of the network and load characteristics.

1) *Performance under Heavy Load*

We repeated the simulation in Section. III with datastore size =40 . Fig. 7 shows the availability of the system (i.e. the request hit ratio) as the load increases. Fig. 8 shows the average hops per request. As is obvious from Fig. 7 and 8, enforced-clustering results in higher availability and lower average hops per request compared to LRU. Also, the cache replacement scheme motivated by the small-world example (enforced-clustering with random shortcuts) significantly outperforms both LRU and enforced-clustering. Fig. 9 illustrates another interesting phenomenon. In this figure, we plot the average number of hops per successful request. The original simple LRU scheme performs much better than enforced-clustering in this metric. However, enforced-clustering with random shortcuts matches the performance of LRU for this metric.

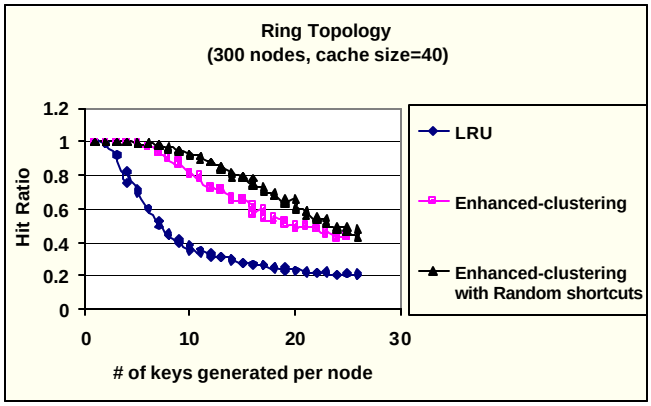


Fig. 7. Availability of Freenet with LRU, enforced-clustering, and enforced-clustering with random shortcuts

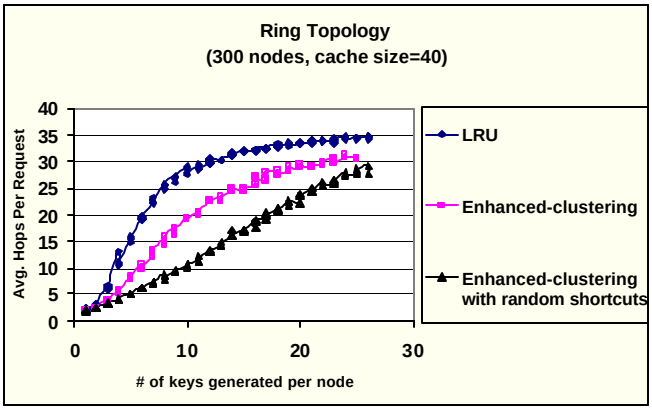


Fig. 8. Average hops per request for Freenet with LRU, enforced-clustering, and enforced-clustering with random shortcuts

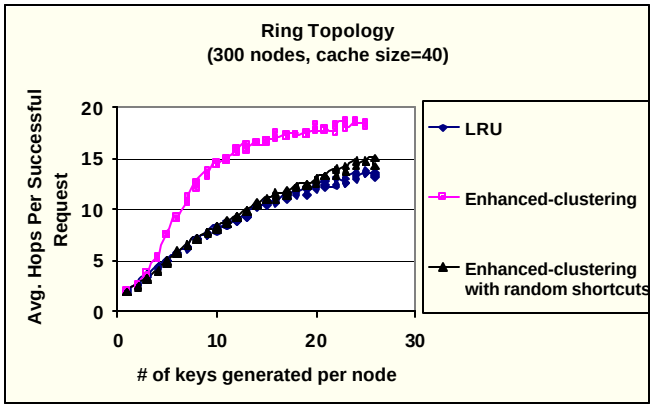


Fig. 9. Average hops per successful request for Freenet with LRU, enforced-clustering, and enforced-clustering with random shortcuts

2) Evolution of Routing Performance with time

To see how network performance evolved with time, we repeated the simulation for using a load of 10 keys per node with 60000 time steps to make sure the network performance was stable at the end of the simulation. Fig. 10 and 11 shows the hit ratio and average hops per successful request in different time phase for those three schemes. Again, we note that enhanced-clustering with random shortcuts resulted in a significantly better hit ratio than LRU while keeping the average number of hops roughly the same as LRU. In addition, only enhanced-clustering with random shortcuts showed an increasing in the hit ratio and a decreasing in Average hops per successful request along with the time simultaneously.

3) Key Distribution in Routing Tables

Finally, we draw the typical key distribution (at the end of the simulation) in Freenet node's routing table under heavy load for enforced-clustering and enforced-clustering with random shortcuts (Fig. 12 & Fig. 13). Fig. 12 shows that the network under enforced-clustering can be modeled as a regular graph. Fig. 13 shows enforced-clustering with random shortcuts does shape the routing table to approximate the one in Fig. 6.

4) Other Simulation Scenarios

We ran the simulations with different initial topologies (ring+random, tree, tree+random, star+random, random), varying number of nodes (300-3000), different values of HopsToLive (40-100) and varying cache sizes (50-200). All simulation results showed qualitatively similar trends.

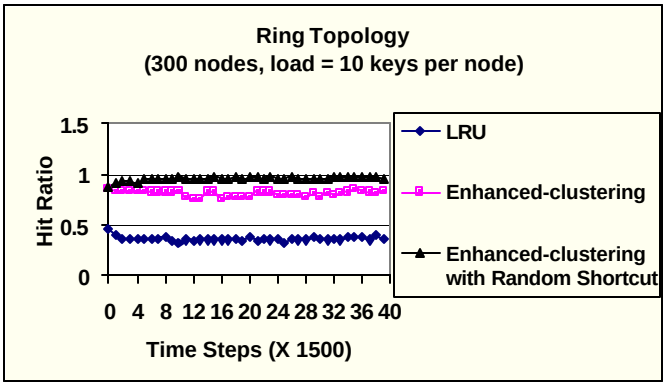


Fig.10. Evolution of Hit Ratio with time for Freenet with LRU, enforced-clustering, and enforced-clustering with random shortcuts

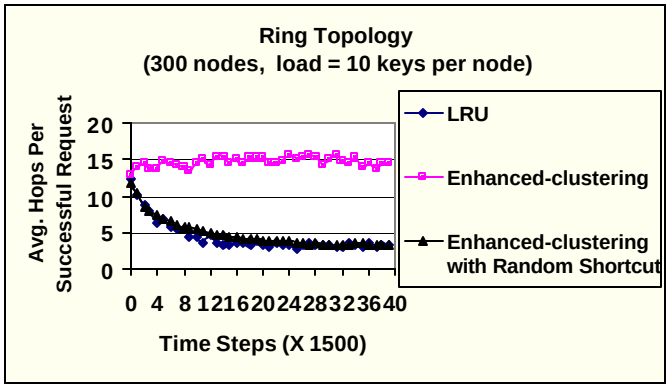


Fig.11. Evolution of average hops per successful request with time for Freenet with LRU, enforced-clustering, and enforced-clustering with random shortcuts

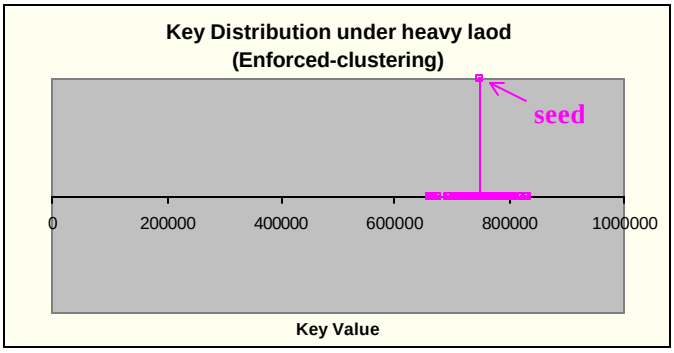


Fig.12. The keys in the routing table cluster around the seed of this node. All keys generated by the node itself are removed in this graph. (in this case average number of keys generated per node =20)

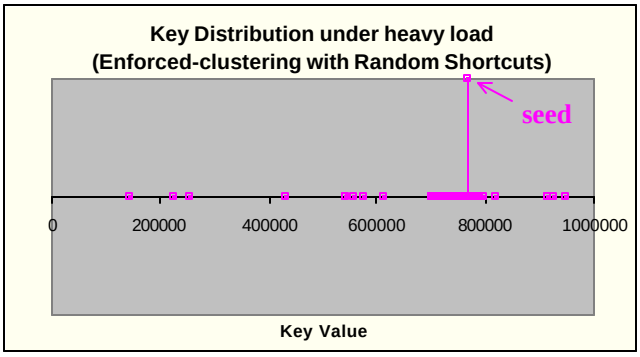


Fig.13. The keys in the routing table cluster around the seed of this node and some random keys distribute in the key space . All keys generated by the node itself are removed in this graph. (in this case average number of keys generated per node =20)

5) Discussion

We now present some additional intuition about why the enhanced-clustering with random shortcuts performs well. Fig. 5 shows that the routing table of Freenet is not very clustered when LRU is used as the cache-replacement scheme. This is analogous to *random graphs*. It is well known that random graphs have small diameters as long as the number of edges is sufficiently large [17]. However, since the edges are drawn at random, it is hard to use local rules to travel from a given node to the node which contains the desired key. Not surprisingly, under high load, LRU has a low hit ratio. Fig. 12 shows that the routing table for Freenet with Enhanced clustering (no shortcuts) is completely clustered. This corresponds to a highly regular graph, where each node is connected to its neighbors. It is very easy to get from a given node to a desired node in such a graph. However, the high clustering implies that each hop travels only a small distance in the key-space resulting in a high number of average hops. Fig.13 shows that the enhanced clustering with random shortcuts results in a small-world like graph. The high clustering in this graph makes it easy to use local rules to arrive at a desired node; at the same time, the random shortcuts sometimes allow us to travel large distances in the key-space using one hop.

Kleinberg showed that for efficient routing, a node needs to choose a shortcut at a distance x with probability proportional to 1/x [12]. This can also be implemented using the following local replacement rule: suppose there are two keys u and v which are contending for being the shortcut keys out of a Freenet node Y. Let x_u and x_v be their distances from the seed key Y. Then we would keep u with probability $x_v/(x_u+x_v)$ and keep v with probability $x_u/(x_u+x_v)$. However, we have chosen to implement the simpler scheme outlined earlier in this section.

It is important to mention that several simple variants of our scheme should also give similar end results (clustering with light randomness). Also, LRU may well have other practical advantages that would override the improvements presented in this section. In addition, our workload may be biased against LRU because of our assumption about the uniformity of access to files. LRU might very well perform reasonably under a different workload where file "popularity" is Zipf distributed, for example. However, the point of the paper is that enhanced-clustering with light randomness enables the system to preserve its small-world property regardless of workload, and this is a desirable design.

V. ANALYSIS

Our analytical results are an attempt to formally demonstrate that the small-world model can lead to good results in an idealized Freenet-like scenario. These results are not hard evidence of the superiority of our scheme. The analysis is a two-step procedure. First, we derive the expected number of hops for a request in an idealized model for Freenet. Then we extend the idealized model by considering the effect of "misleading" links and show that the performance (polylogarithmic number of hops) can be kept as long as a polylogarithmic-sized routing table/datastore is provided in Freenet.

A. An Idealized Network Model

We consider an idealized model of Freenet which assumes that the seeds are chosen so that they are distributed evenly in the key space and the routing tables are shaped to fit the small-world hypothesis, i.e., the routing tables at each node have the structure shown in Fig. 6. Assume each node has a (2K+1)-files-sized data store and a (2M+1)-entries routing table. M should be no less than K since the routing table should maintain at least the pointers to the (2K+1) files in the data store. In the idealized model each node X caches the files for the 2K keys centered at Seed(X) and a randomly chosen key. The routing table of X will keep entries for the 2M keys centered at Seed(X) and the randomly chosen key. Fig.14 shows how the links are created from one node to another. The arrow from a key to a node means this node contains this key in its data store and therefore claims itself as the source of this key. The arrow from a node to a key means this node has an entry for this key in its routing table. A link exists from nodes X to node Y iff a transit key u exists between them and is called X→u→Y. For example, in Fig.14 Node X has a link to Y via the transit key v. When searching for a key k, a link X→u→Y is called a "misleading" link if u is the closest key to k in the routing table of X but Y can't find a closer key v to k than u so that Y can forward the request to v's corresponding node. "Misleading" links may exist in Freenet but we will not consider the effect of "misleading" links until the next subsection. Therefore, an idealized model from the node level is shown as Fig. 15. In Fig. 15 the solid links stands for the local contacts between each pair of neighbors via their overlapped routing tables and the dotted links stand for the random shortcuts the nodes choose. Each node has two pointers to its two neighbors and a link pointing to a node randomly chosen in the network. We assume that all nodes forget the initial inserters of the files after a long time. The source of any file(key) is known to everyone as the current holder of this file.

Theorem 1: In the idealized network model without the effect of "misleading" links, if each node x chooses its random shortcut so that the random shortcut has an endpoint y with probability proportional to 1/d where d = $|s_x$

- s_y|, *then the expected number of hops required to find a document in this idealized system is* $19\log^2 n$ *using Freenet's search algorithm. Here n is the number of nodes in the system.*

Proof: The idealized model without the effect of "misleading" links is actually a one-dimensional version of Kleinberg's model in [12]. The result follows from a similar theorem in [12]. ■

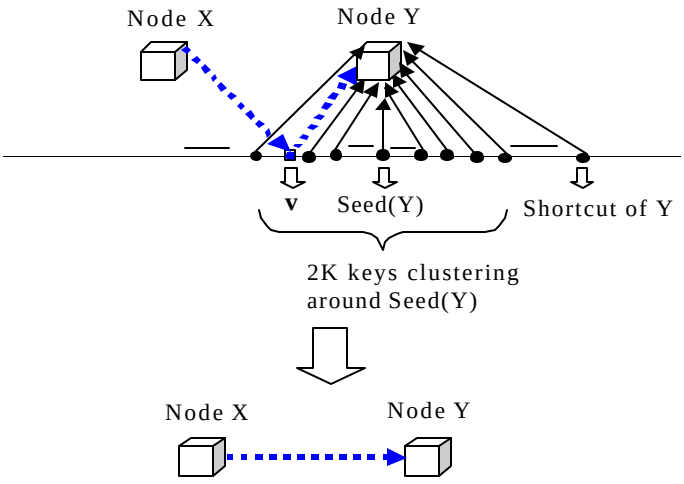


Fig.14. Node-to-node links are created by transit keys.

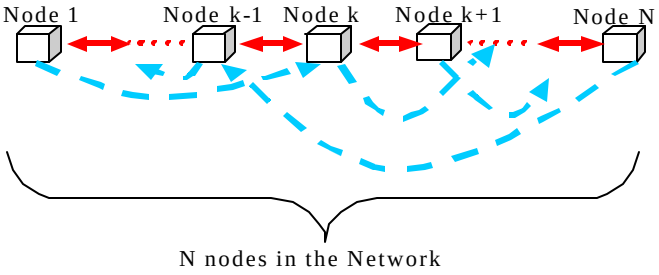


Fig. 15. An idealized model of Freenet from the node level. The long edges correspond to shortcut connections.

B. *Idealized Model with Misleading Links*

Theorem 1 holds only if the distance from the request to target k decreases strictly in each step during the searching. However, this condition doesn't exist when there are misleading links in Freenet. In this subsection we will consider the possibility that a "misleading" link will be chosen at each step of a search in Freenet. We assume that while searching for key k, some node X forwarded the request to Y. The node X must have had an entry of the form ⟨k', Y⟩ in its routing table. k' must be one of the keys in Y's data store. Let us assume that k' can be any of the (2K+1) keys which are present in the data store of Y with equal probability. Clearly, Y will now find a tuple ⟨k*, V⟩ in its routing table such that k* is closest to k, and then forward the request to node V. But there are two cases in which Y can't find such a node V:

1. k' is the shortcut of Y. In this case all other 2K keys may be far away from the target key k (shown in Fig.16 a);
2. k' is the closest key to k in Y's 2K clustered keys (shown in Fig.16 b).

Then, the possibility that a "misleading" link is chosen is equal to 2/(2K+1) < 1/K. Next we will prove that Freenet's performance (polylogarithmic number of hops) in Theorem 1 can be kept even under the effect of "misleading" links as long as a polylogarithmic-sized routing table/datastore is provided in each node. The large constants in this theorem are an artifact of our analysis. Even with these large constants, we believe this analysis is interesting since it points towards the correct ballpark performance.

Theorem 2: In the idealized network model considering the effect of "misleading" links, if K= $76\log^2 n$ *and each node x chooses its random shortcut so that the random shortcut has an endpoint y with probability proportional to 1/d where d =* $|s_x - s_y|$*,, then the expected number of hops required to find a document in this idealized system is O(*$\log^2 n$*) using Freenet's search algorithm. Here n is the number of nodes in the system.*

Proof: From Theorem.1 we know a search will end in at most $19\log^2 n$ steps on the average when there is no effect of "misleading" links. We count $38\log^2 n$ steps as one run. If no "misleading" links are encountered during a run, then using Theorem 1 and Markov's inequality[18], we know that the probability of finding the key in that run is at least 1/2. Let q be the probability of encountering no misleading link during a run, and let p be the probability that that run is successful under the effect of "misleading" links. Then p ≥ (q – 1/2). Also

$$q \geq (1-1/k)^{38\log^2 n}$$
$$\geq 1/\sqrt{e} = 0.606$$
(n>>10)

Therefore, p ≥ 0.106. Let X denotes the total number of runs to find a document. We have

$$E[X] = \sum_{i=1}^{\infty} \Pr[X \geq i]$$
$$\leq \sum_{i=1}^{\infty} (1-p)^{i-1} = 1/p \approx 9.4$$

Therefore, expected number of hops to find a document is at most

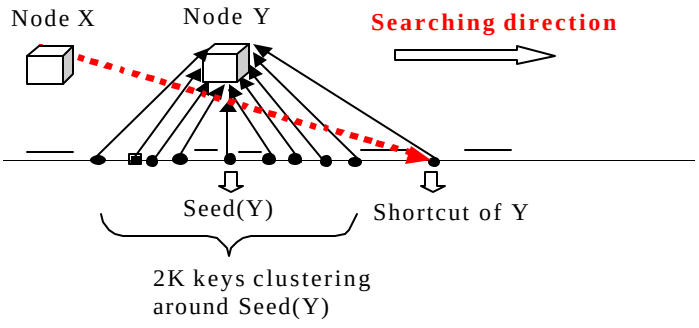
$$9.4*(38\log^2 n) = O(\log^2 n).$$

■

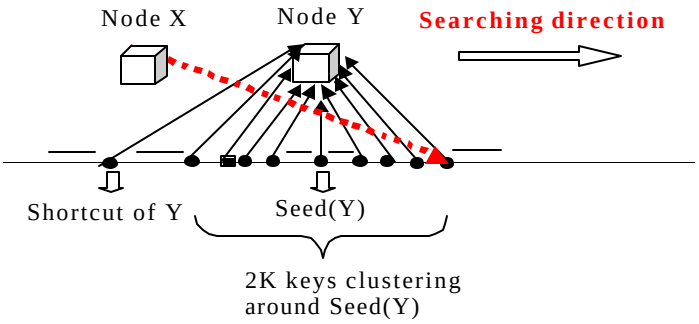
Theorems 1 and 2 state that, our replacement policy motivated by the small-world model gives a polylogarithmic number of hops on the average in our idealized model, both with and without "misleading" links. Here there is a requirement that the random shortcut be chosen from a specific distribution (called inverse r^{th}-power distribution in [12]). It is possible to satisfy this requirement in our scheme by using different values of p depending on the distance of the candidate shortcut key from the seed key. The datastore (and routing table size) required is also polylogarithmic in the

size of the Freenet system. Of course the size of the data store should be at least large enough to make sure that every key has a copy in some node.

To summarize, in this idealized model, a carefully configured unstructured system gives the same ballpark performance (i.e. polylogarithmic) in terms of the size of the data store/routing table and the average number of hops as the structured schemes. If Theorem 2 continues to hold in more realistic settings, then it would be an important consideration in the design of peer-to-peer networked systems.



(a) A request is passed from X to Y through the shortcut of Y



(b) A request is passed from X to Y through the rightmost key of Y

Fig. 16. Two cases in which the request is forwarded through a "misleading" link.

VI. CONCLUSIONS

In this paper we sketched some preliminary work we did towards improving the performance of unstructured systems such as Freenet by using intuition from the small–world model. We first gave a brief description of the Freenet protocol. We then sketched the main idea behind the small–world model and explained how we could use intuition gained from this model to influence Freenet performance. We presented a new but simple cache replacement scheme: enhanced–clustering with light randomness. Our simulations showed a significant increase in availability and a significant decrease in the average number of hops when we used the enhanced–clustering caching with random shortcuts rather than LRU or enhanced–clustering caching without random shortcuts. It is important to note that this change did not involve any modifications to the Freenet protocol, just to local user behavior when the datastore gets filled. Finally, we analyzed the latency of Freenet (with the modified cache replacement scheme) in an idealized model and proved that the average time to find a key in such a model is just $O(\log^2 n)$ where n is the number of nodes in the system. In addition to being interesting in their own right, the results sketched in this paper also illustrate how theoretical techniques and insights can improve the performance of peer–to–peer systems.

One specific open problem motivated by this work is to find the best value of p as a function of the network and load characteristics. Also, it would be interesting to do a more complete analysis of Freenet as a stochastic system as opposed to our simple idealized model. More generally, there is a need to take a principled look at the properties of various algorithms proposed in the peer–to–peer literature. Peer–to–peer networks have a rich theoretical structure, and sophisticated algorithmic techniques have been employed in the systems currently under development [3][4][5]. Some of the rich structure of this problem was exemplified by the pioneering work of Plaxton et al. [14], where they give an elegant scheme to achieve optimum latency in networks which follow power-law expansion [15]. This has recently been extended to all graphs, but with weaker performance guarantees [19]. Some of the specific questions that need to be addressed are understanding the fundamental bounds on the performance of peer–to–peer systems, designing algorithms that allow these systems to perform at peak availability and minimum latency, and understanding the role that the small-world model might play in improving the performance of the peer–to–peer systems.

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their useful comments.

REFERENCES

[1] Napster. http://www.napster.com.
[2] Gnutella. http://www.gnutella.co.uk.
[3] S. Ratnaswamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content- addressable network. ACM SIGCOMM, 2001.
[4] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A peer-to-peer lookup service for internet applications. ACM SIGCOMM, 2001.
[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.
[6] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system in designing privacy enhancing technologies. International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, 2001.

[7] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. Proceedings of ACM Sigmetrics, 2000.

[8] P. Francis. Yallcast: Extending the internet multicast infrastructure. Unrefereed Report, http://www.yallcast.com/docs/index.html, 1999.

[9] A. Goel and K. Munagala. Extending greedy multicast routing to delay sensitive applications. ACM Symposium on Discrete Algorithms (short abstract). Also, to appear as an invited paper in the special issue of Algorithmica on Internet Algorithms, July 1999.

[10] D.J. Watts. Small-worlds: The Dynamics of Networks between Order and Randomness. Princeton University Press, 1999.

[11] J. Kleinberg. Navigation in a small-world. Nature, 406, 2000.

[12] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. Cornell Computer Science Technical Report 99-1776, 2000.

[13] T. Hong, Performance. In Peer-to-Peer: Harnessing the Power of Disruptive Technologies, ed. by A. Oram. O'Reilly and Associates: Sebastopol, CA (2001)

[14] C.G. Plaxton, R. Rajaraman, and A.W Richa. Accessing nearby copies of replicated objects in a distributed environment. Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures, Newport, Rhode Island, pages 311--320, June 1997.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. ACM SIGCOMM, 1999.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of `small-world' networks, Nature 393, 440--442 (1998).

[17] B. Bollobas, Random Graphs. Academic Press, London, 1985

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

[19] R. Rajaraman, A. W. Richa, B. Vöcking, G. Vuppuluri. A Data Tracking Scheme for General Networks. Thirteen ACM Symposium on Parallel Algorithms and Architectures, 2001.