The Effect of Replica Placement on Routing Robustness in Distributed Hash Tables *

Cyrus Harvesf and Douglas M. Blough
Georgia Institute of Technology
School of Electrical and Computer Engineering
Atlanta, GA 30318
{charvesf, dblough}@ece.gatech.edu

Abstract

To achieve higher efficiency over their unstructured counterparts, structured peer-to-peer systems hold each node responsible for serving a specified set of keys and correctly routing lookups. Unfortunately, malicious participants can abuse these responsibilities to deny access to a set of keys or misroute lookups. We look to address both of these problems through replica placement. Using Chord as an example, we present an equally-spaced replication scheme and prove that it can be tuned to produce any desired number of disjoint routes. To be specific, we prove that d disjoint routes can be produced by placing 2^{d-1} replicas around a fully populated Chord ring in an equally-spaced fashion. In this situation, we also prove that there exists a route to at least one replica, which contains only uncompromised nodes, even if an attacker controls more than a quarter of the contiguous identifier space in the system. Simulation experiments demonstrate that this scheme performs better than previously proposed replica placement schemes in rings that are sparsely populated, populated in clusters, or populated partially by compromised nodes.

1. Introduction

Structured peer-to-peer networks provide the benefits of efficiency, scalability, resilience, and self-organization when building distributed applications. These systems achieve efficiency by utilizing their structure to reduce redundant operations. With this structure comes added responsibility for each node participating in the system. For instance, Chord [16] and other distributed hash table (DHT) implementations depend on each node to efficiently route lookups. Therefore, each node has the power to route and *misroute* lookups. Malicious routing is not the only vulnerability; adversaries may act upon several attack vectors. Before DHTs can be applied as real world solutions in areas like the domain name system and voice over IP systems, these vulnerabilities must be addressed.

In our work, we consider two such attacks: malicious routing and storage and retrieval attacks. In a storage and retrieval attack, an adversary may seize control of a key or a set of keys by selecting an appropriate node identifier. The most commonly accepted defenses to these attacks are multipath routing and replication, respectively. Our solution integrates these two approaches into a replica placement scheme that can be tuned to produce a desired number of disjoint routes.

2. Background

Sit and Morris [13] identify the major threats to structured peer-to-peer systems to be:

1. Routing attacks, in which a malicious node may misroute lookups or attempt to corrupt routing tables;

2. Storage and retrieval attacks, in which a malicious node may claim that a key it serves does not exist; and

3. Miscellaneous attacks, in which a malicious node may act arbitrarily to tamper with the correct operation of the system. These attacks include Sybil and Eclipse attacks [5, 12] as well as denial of service (DoS) attacks [7, 15].

Our work mitigates the first two threat types. As suggested in [14], routing attacks can be mitigated with alternative lookup paths. To this end, they propose the notion of *independent lookup paths*. Two routes are said to be independent if they share no common node other than the

*This research was funded in part by the National Science Foundation under Grant ITR-NHS-0427700.

source and destination nodes. Independent routes help improve routing robustness, but do not address storage and retrieval attacks because the routes share a common destination node. We define *disjoint routes* to be routes that share no common node other than the source. This is a more robust notion of alternative paths.

To address storage and retrieval attacks, we use replication as recommended in [13]. The novel contribution of our approach is that the replica placement naturally produces disjoint routes without significant modification to the underlying peer-to-peer system.

In order for disjoint routes to improve the robustness of our system, we require that the keys stored in the system be self-verifying; that is, we assume that a key's integrity can be verified by the client. This is necessary to detect when a node returns an incorrect entry. For types of data where self-verification is not natural, we can generate key identifiers by hashing the key object as in [4]. The client can verify the key's integrity by comparing its hash with the key identifier. Castro et al. [2] discuss self-verifying data as well as a method for managing mutable self-verifying data.

Under these assumptions, our system can tolerate attacks by up to $d - 1$ malicious nodes acting arbitrarily, where d is the number of disjoint routes. We also prove, under the same assumptions, that the approach can tolerate a *run* of contiguous nodes controlled by an adversary covering more than one-quarter of the identifier space. To our knowledge, this is the first approach to peer-to-peer routing for which properties such as these have been formally proven.

3. Related Work

To our knowledge, no work has been done to use replica placement to mitigate malicious routing behavior in peer-to-peer networks. However, significant work has been done in the areas of peer-to-peer routing security and replica placement. In the following subsections, we discuss these previous works and contrast them with our solution.

3.1. Peer-to-Peer Routing Security

Many works have looked to improve the security and robustness of peer-to-peer routing. Many approaches have been centered around the notion of alternative paths [14].

Artigas, et al. [1] use a multipath concept to secure routing. They propose Cyclone, an equivalence-based routing scheme that is built upon an existing peer-to-peer structured overlay. Independent lookup paths can be created by routing along different equivalence classes. The key difference between Cyclone and our work is that Cyclone creates independent lookup paths rather than disjoint paths. Since the paths created do not differ in the destination node, Cyclone will not prevent storage and retrieval attacks. Our work produces disjoint routes to several appropriately placed replicas, thereby preventing a limited number of attacks of this kind. Furthermore, Cyclone requires additional routing overhead per node and employs a complex routing scheme. We use a replica placement scheme that automatically produces disjoint routes without significant modification to the underlying routing mechanism.

The notion of independent paths has been considered in other works as well. In an effort to provide message confidentiality, [11] suggest that messages be split, encrypted, and sent to the destination along independent paths. Empirically, [11] determines the finger table offsets to minimize route overlap with high probability. We will show analytically that our placement scheme creates disjoint routes.

Castro et al. [2] propose a secure routing primitive using replication and two routing mechanisms. They combine efficient, locality-based routing with constrained routing mechanisms to find diverse routes to the replica set in the event of routing failure. Rather than modifying the underlying routing mechanism, we improve routing robustness by carefully placing replicas. Using the robustness properties we prove herein for equally-spaced replica placement, we believe this technique can be combined with Castro's routing approach to achieve even greater security. Thoroughly investigating the combination of optimized replica placement and diverse routing is an interesting topic for future research.

3.2. Replica Placement

Replica placement has long been studied in realms outside of peer-to-peer systems. As more work has been done within peer-to-peer systems, it has become clear that replica placement can be used to manage load and satisfy capacity constraints while improving the quality of service in the system. Many studies have compared the performance of several placement schemes in terms of quality of service, availability, and time to recovery [3, 6, 8, 9]. However, to our knowledge, none have considered routing robustness as it relates to replica placement.

Castro et al. [2] discuss four well-known distributed hash tables–CAN, Tapestry, Pastry, and Chord–within the context of secure routing. These systems use two basic types of replica placement: neighborhood-based and random. Pastry and Chord place replicas at the node locations "closest" to the node storing the master replica. CAN and Tapestry use hash functions to randomly map replicas within the key space. A similar hash function-based, random replica placement and enumeration scheme is discussed in [18]. We propose an equally-spaced replica placement scheme that outperforms these existing schemes in terms of routing robustness.

4. Equally-Spaced Replication

In this section, we discuss a replica placement scheme that produces disjoint routes for each replica set. The simplest method for generating disjoint routes is to ensure that the routes are contained within non-overlapping segments of the ring; this is the premise of our work. Although we discuss equally-spaced replica placement in the context of Chord, it can be applied to distributed hash tables that utilize the prefix matching routing technique [10].

As replication is implemented in [16], keys are replicated in a chain of nodes starting at the key's home node. To access a correct replica, the lookup begins at the first node in the chain and iterates down the chain until a correct replica is found. It is clear that the routes to each of the replicas differ only in the final hop. An adversary could very easily deny access to the entire replica set by controlling a single node in the overlapping portion of the routes.

In our approach, we place replicas at equally-spaced locations over the entire ring. Rather than using the same route for each replica, we initiate separate lookups for each object in the replica set. Our placement scheme generates disjoint routes, which improves routing robustness.

4.1. Evaluation of Disjoint Routes

The proposed solution is simple: replicas are assigned equally-spaced identifiers starting at the master key identifier and proceeding in a wrap-around fashion over the entire identifier space. For example, consider a Chord ring of size 256 with a replication degree of 4. The four replicas of key 71 will be assigned identifiers 71, 135, 199, and 7. Note that the replicas are equally-spaced; that is, the difference between consecutive replica identifiers is equal to the total identifier space size divided by the replication degree (e.g. $135 - 71 = 64 = \frac{256}{4}$). We claim that this replication scheme produces a predictable number of disjoint routes. To prove this claim, we first show that routes originating from a common query node that differ in the first hop must be disjoint.

Lemma. *Routes originating from a common query node that differ in the first hop are disjoint.*

Proof. Consider a query node q and two routes originating at q destined for keys k_1 and k_2[1]. Suppose that the first hops in the routes to these keys are $q+2^i$ and $q+2^j$, respectively, where $i \neq j$. According to the definition of the Chord protocol, $k_1 \in \left[q+2^i, q+2^{i+1}\right)$ and $k_2 \in \left[q+2^j, q+2^{j+1}\right)$. Furthermore, each hop in these routes must reside in their respective intervals. Therefore, the routes from q to k_1 and k_2 are disjoint. □

[1] All identifiers are ordered on an *identifier circle* modulo n.

This lemma, though straightforward, provides insight into the premise behind our solution. To produce disjoint routes from a common source, the first hops in each route must be different. The first hop in a route is determined by the distance between the source and destination nodes. Therefore, if we place replicas at carefully chosen locations, we can produce disjoint routes.

We prove our claim within the context of a *full Chord ring*. We define a full Chord ring to be a Chord ring wherein all possible identifiers are represented. In other words, for any key k, k will be located at the node with identifier k.

Theorem. *To produce d disjoint routes to a key k from any query node in a full Chord ring of size n with equally-spaced replicas ($d \leq \log_2 n + 1$), k must be replicated at exactly 2^{d-1} locations in a wrap-around fashion starting at k.*

Proof. (by induction) In a full Chord ring of size n with $d = 2$, k is replicated at nodes k and $k + \frac{n}{2}$. Consider a query node q. Let k_1 be the replica in the interval $\left[q, q + \frac{n}{2}\right)$ and k_2 be in the interval $\left[q + \frac{n}{2}, q\right)$. (Since k_1 and k_2 are equally-spaced in the ring, they must reside on different halves of the ring.) The first hop of the route from q to k_1 must be in $\left[q, q + \frac{n}{2}\right)$ and the first hop of the route from q to k_2 must be $q + \frac{n}{2}$. Thus, the routes from q to k_1 and k_2 are disjoint.

Assume 2^{d-1} equally-spaced replicas produce d disjoint routes to k. Let I be the largest interval $\left[q, q + 2^i\right)$ that contains exactly one replica of k. Since replicas are equally-spaced and $d < \log_2 n + 1$ (i.e. there are fewer replicas than nodes in the system), the interval I exists. Increasing the replication degree to 2^d guarantees that there are two replicas in I. Since the 2^{d-1} replicas that formed d disjoint routes are a subset of the new set of replicas, it is enough to show that the two replicas in I produce disjoint routes. Let k_1 and k_2 be the identifiers of these two replicas such that $k_1 \in \left[q, q + 2^{i-1}\right)$ and $k_2 \in \left[q + 2^{i-1}, q + 2^i\right)$. The first hop to k_1 is in $\left[q, q + 2^{i-1}\right)$ and the first hop to k_2 is $q + 2^{i-1}$. Therefore, the routes to k_1 and k_2 are disjoint. □

Although these claims are proven for a full Chord ring, we hope to demonstrate that these concepts can be applied to sparsely populated Chord rings with similar performance. A formal analysis of the performance in sparsely populated rings is left for future work. However, extensive simulation results in sparse rings are provided in Section 5.2.

4.2. Toleration of Runs

In this section, we introduce the concept of a *run*, the way in which a run can be used to disrupt the system, and the degree to which equally-spaced replication can tolerate runs. We define a run of length l to be a contiguous set of l

nodes from the Chord ring. Equivalently, the run of length l starting at node m is denoted in set notation by $[m, m+l)$.

As indicated in [16], an adversary can create imbalance in the distribution of nodes in the ring by appropriately selecting identifiers. In the worst case, an adversary can take control of a contiguous sequence of identifiers or, using our terminology, a run of nodes.

We claim that equally-spaced replication can tolerate adversarial runs with bounded length. Before proving the tolerable length of a run, we provide some intuition of how a run may be used to disrupt routing in the ring. Consider a query node q. An adversary can reduce the number of replicas reachable from q by half by controlling the node $q + \frac{n}{2}$. This is because all of the replicas in interval $[q + \frac{n}{2}, q)$ are routed through the node $q + \frac{n}{2}$ (half of the replicas are in this interval). If the adversary controls a run of nodes ending at $q + \frac{n}{2}$, he can control a larger number of replicas.

Theorem. *A full Chord ring of size n with 2^{d-1} ($d > 2$) equally-spaced replicas can tolerate any adversarial run of length $1 + n(\frac{1}{2} - \frac{1}{2^{d-1}})$.*

Proof. (by induction) Consider a query node q. For $d = 3$, there exists a replica $k \in \left[q, q + \frac{n}{4}\right)$. (There are four replicas; one must reside in the first quadrant of the ring starting at q.) If we assume that the adversary has control of the node $q + \frac{n}{2}$ (which is the worst case), then we must ensure that k is not in the run. Thus, the maximum length run is $[q + \frac{n}{4}, q + \frac{n}{2} + 1)$, which has length $\frac{n}{4} + 1$ or $1 + n(\frac{1}{2} - \frac{1}{2^2})$.

Assume that the longest tolerable run in a full Chord ring with 2^{d-1} equally-spaced replicas has length $1 + n(\frac{1}{2} - \frac{1}{2^{d-1}})$. Consider a query node q. If we assume that the adversary takes control of the node $q + \frac{n}{2}$, then he does not control any nodes in the interval $[q, q + \frac{n}{2^{d-1}})$. There must exist at least one replica k in this interval. If we double the replication degree to 2^d, then there are two replicas in this interval separated by $\frac{n}{2^d}$. Thus, the length of the tolerable run increases by $\frac{n}{2^d}$ to $1 + n(\frac{1}{2} - \frac{1}{2^d})$. □

Note that with four equally-spaced replicas, an adversary may control of a run of more than a quarter of the nodes in the system and there will be a route to one replica for every possible query. As the replication degree approaches the identifier space size, the maximum tolerable length approaches one-half the identifier space.

4.3. Per-key Replication

Thus far, our discussion has assumed a fixed replication degree for every key inserted into the ring. However, it is oftentimes the case that certain objects inserted into the system are more critical than others. It would be desirable to replicate these objects to a higher degree and maintain the benefits of equally-spaced replication.

The assumption of a fixed system-wide replication degree is important because it allows any node to compute the locations of all replicas for a given key. However, this is not a necessary condition. For example, suppose we have a Chord ring wherein the most critical objects should be stored at 16 equally-spaced locations throughout the ring. We can replicate objects that are less critical at 8, 4, or 2 locations in the ring. Since the replicas are equally-spaced, we can guarantee that the actual replica locations will be a subset of the 16 possible locations regardless of the key's actual replication degree. Note that this technique reduces the replication effort for less critical objects without increasing the query routing cost.

4.4. Implementation

To uniquely identify each replica, we use a *key identifier pair* (k, v), where k is the *key identifier* and v is *virtual key identifier*. For each replica, v gives the location of the master key. By definition, the master key k is denoted by the pair (k, k).

When a key k is inserted into a Chord ring of size n with replication degree d, we first compute the key identifier pairs for each replica: $(k, k), (k+\frac{n}{d}, k), (k+\frac{2n}{d}, k), \ldots (k+\frac{(d-1)n}{d}, k)$. For example, consider a Chord ring of size 256 with replication degree 4. When the key 71 is inserted, we compute $(71, 71)$, $(135, 71)$, $(199, 71)$, and $(7, 71)$.

Once the key identifier pair for each replica is computed, we use the traditional Chord key insertion mechanism to insert the replicas. That is, we perform a lookup for each key identifier and store the replica at those locations. In the example above, we lookup keys 71, 135, 199, and 7 and store the replicas at their respective locations.

Next, the Chord lookup primitive must be modified to accommodate the new replication scheme. When a node is queried for a key, the query node must first compute the locations of all replicas (using the known ring size and replication degree). The key identifier is used to route to the replicas. Once a replica's home node is found, the key identifier pairs are compared to return the appropriate replica.

It is worth noting that no additional finger table entries are required to route to the replicas. In addition, if the query node dispatches the lookups for the entire replica set simultaneously, there may be an improvement in performance because the query node can return the first correct response received (which may have returned along a route shorter than the route to the master key). However, if the added load of the extra lookups puts strain on the system, the performance may improve only slightly or even degrade. A study of the effect of these extra lookups on system performance is a topic for future work.

Finally, when nodes join or leave the ring, traditional Chord node join and leave mechanisms can be used by sim-

ply ignoring the virtual peer identifiers in each key identifier pair. Note that the traditional Chord replica placement requires modification to the node join and leave mechanisms. To maintain the replication degree, replicas will need to be shifted for every node join or leave.

5. Experiments

5.1. Experimental Setup

A simulator was designed to model Chord rings at message-level detail; that is, the simulator computes routes from any query node to any key using finger tables. The simulator was used to measure the performance of replication schemes over uniform and clustered node distributions. To simulate a system with little or no malicious activity, a uniform distribution was used to assign node identifiers.

One potential attack on equally-spaced replication is to cluster the node identifiers, leaving unpopulated gaps in the identifier space. With this distribution of identifiers, it is possible that two or more replicas may be managed by a single node, which eliminates one or more possible disjoint routes. We model clustering by selecting an identifier for the cluster mean and randomly sampling node identifiers from a Gaussian distribution centered at the cluster mean. In addition to varying the number of clusters, we can vary the size of overlap by tuning the variances of the distributions. The simulation parameters are summarized in Table 1.

Symbol	Parameter
N	Identifier Space Size
n	Number of Nodes
C	Number of Clusters
σ	Cluster Width (standard deviation)

Table 1. Simulation Parameters.

Finally, we compare the performance of equally-spaced replication to other schemes. Namely, we consider the random placement of replicas, a variant of the replication scheme used in [16], and a simple spaced replication scheme. We implement the same placement used in [16]; that is, we place replicas at the chain of nodes starting at the key's home node. However, rather than using a single lookup to reach all replicas, we perform independent lookups for each replica, which may slightly increase the number of disjoint routes.

For the spaced replication scheme, we introduce a parameter s that can be used to indicate the spacing between replica identifiers in the chain. Replicas of the key k are placed the locations $k, k+s, k+2s, ..k+(r-1)s$, where r is the replication degree. Unlike the Chord replication variant, spaced replication may result in replicas being co-

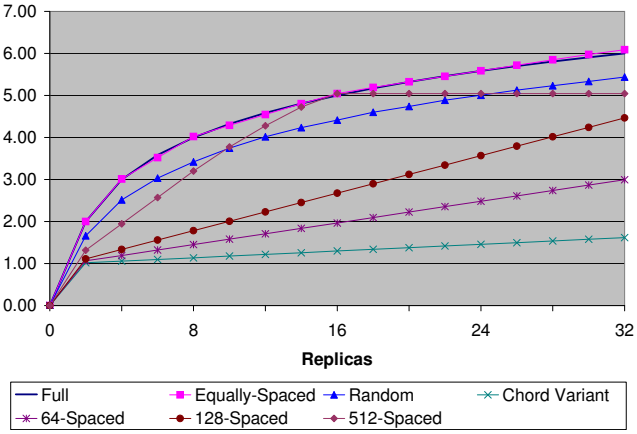

Figure 1. Disjoint Routes for Uniformly Distributed Nodes ($N = 8192, n = 512$).

located for small spacings. Note that when $s = \frac{N}{r}$, this scheme is equivalent to the equally-spaced scheme.

To manage our variant of Chord replication and the random placement scheme, we maintain a mapping of each key to its replica locations. More sophisticated techniques are necessary to efficiently manage the mapping. Chord avoids this complexity by using a single route for all replicas. For equally-spaced replication, replica locations can be computed directly from the key identifier.

5.2. Experimental Results

In each experiment, we consider 10 random node distributions. For each distribution, 25 keys are selected at random and routes are computed from every node in the ring to each replica set. Each data point represents the average over a total of the 250 trials.

The first series of experiments conducted was to measure the performance of each replication scheme with a uniformly populated Chord ring. A Chord ring with $N = 8192$ was modeled with a 6.25% load (or $n = 512$). The average number of disjoint routes per key are shown in Figure 1. The 95% confidence intervals were computed to be less than 0.01 routes for the experiments conducted. Therefore, these results are representative of the relative performance of the schemes tested.

The series marked "Full" reflects the number of disjoint routes expected for a full Chord ring (d for 2^{d-1} replicas). It is clear that equally-spaced replication approximates this curve and outperforms all other replication schemes. A random replication scheme performs well compared to other methods and is only about 10% worse than equally-spaced replication. This is because replica locations are selected at random from a uniform (equally-spaced) distribution. As the number of replicas increase, random repli-

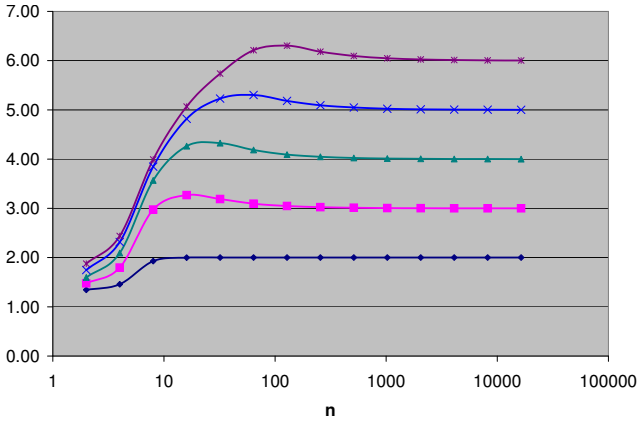

Figure 2. Disjoint Routes with Increasing Uniform Load for 2, 4, 8, 16, and 32 Equally-Spaced Replicas ($N = 2^{20}$).

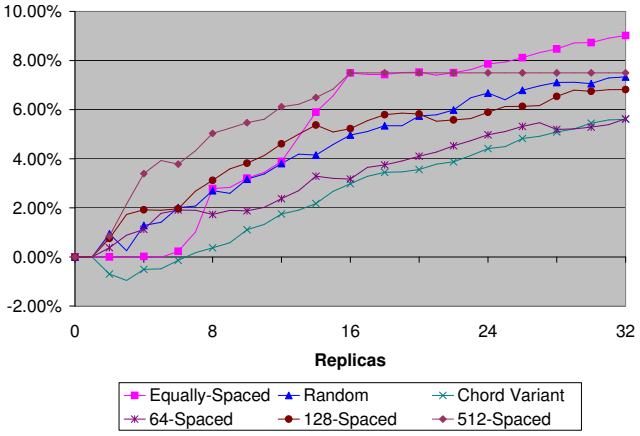

Figure 3. Percent Error for Clustered Nodes ($N = 8192, n = 512, C = 4, \sigma = 200$).

cation will better approximate equally-spaced replication. However, as mentioned above, random replication may introduce added implementation complexity to store the unpredictable replica locations.

Of the replication schemes tested, the Chord replication variant performs worst in terms of the average number of disjoint routes. As expected, the Chord variant does not significantly improve robustness against routing attacks with increasing replication degree.

The remaining series show the increase in performance in using spaced replication with spacings of 64, 128, and 512 between replica identifiers. The number of disjoint routes increases linearly with the number of replicas up to $\frac{N}{s}$ replicas, where s is the spacing. At this point, the optimal number of replicas (as determined by equally-spacing) is reached and no additional disjoint routes can be generated. Once the replication degree surpasses this optimal value, the replicas begin to wrap-around and overlap so that no new disjoint routes are created. This implies that to achieve the best performance, the spacing between replicas should vary with the replication degree, which is precisely what the equally-spaced replication scheme does.

To measure equally-spaced replication performance in sparsely populated Chord rings, N was fixed and n was varied. The results are depicted in Figure 2. For these experiments, N was fixed at 1048576 (a 20-bit identifier space). n is shown on a logarithmic scale to better reflect the rate of convergence. Equally-spaced replication converges to the expected full Chord ring results for $n > 1000$ or about a 0.1% load. These results scale well with network size; to achieve a near-theoretical number of disjoint routes for a 32-bit identifier space, the required load is less than 0.1%. Clearly, equally-spaced replication can achieve near-ideal performance for very sparsely populated Chord rings.

There does seem to be one anomaly in the convergence of the number of disjoint routes; that is, the number of disjoint routes increases above the theoretical value before converging. This is because a small fraction of the replica sets produces one additional disjoint route. This additional disjoint route comes from the replica which precedes the query node in the ring. Typically, when the ring is moderately populated, this replica will be managed by a node that precedes the query node. However, in a sparsely populated ring, this replica may be managed by the query node itself. We count this scenario as an additional route although no routing is actually performed. However, clearly, this replica can be used to satisfy the lookup and should be considered to improve routing robustness.

To model the population distribution that may result from the collusion of several malicious nodes, a series of experiments were run on clustered rings. To model a clustered distribution, four node identifiers were randomly selected as cluster means such that the clusters are non-overlapping and unpopulated gaps exist in the identifier space.

Clustering did not affect the relative performance of the schemes tested; however, there was a decrease in the overall performance. To determine which schemes are most resistant to clustering, we computed the percent error based on the difference between the number of disjoint routes for uniformly distributed and clustered nodes (Figure 3).

For a small number of replicas, equally-spaced replication is resistant to the effects of clustering. To have a noticeable effect, two or more replicas must fall into an unpopulated gap in the identifier space so they are collocated. In the case of equally-spaced replication, the spacing must be smaller than the largest gap size. For eight replicas, the spacing is 1024 for $N = 8192$. Using two standard deviations to capture 95% of the nodes within each cluster, we can estimate the gap size to about 1200 for $\sigma = 200$, which

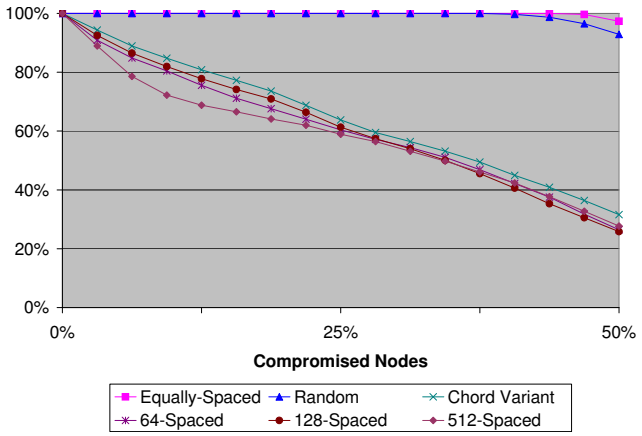

Figure 4. Probability of Routing Success with Compromised Runs for a Replication Degree of 4 ($N = 2^{20}, n = 1024$).

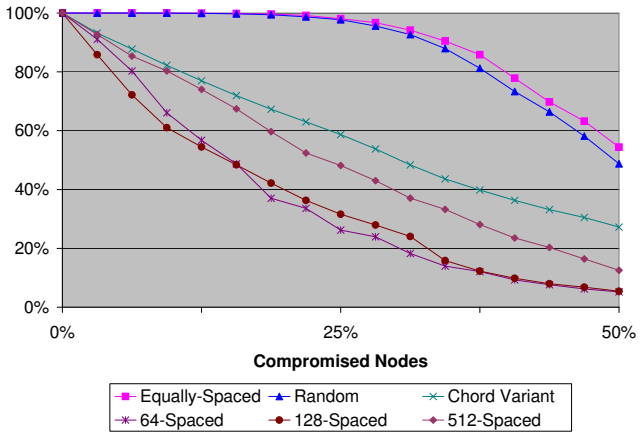

Figure 5. Probability of Routing Success with Randomly Compromised Nodes for a Replication Degree of 4 ($N = 2^{20}, n = 1024$).

is greater than the spacing. This correlates well with the data; the error does not significantly increase until a replication degree of eight is reached. Therefore, for a small number of replicas, where the spacing is large compared to the gap size, equally-spaced replication can resist the dense clustering of nodes.

When the number of replicas increase, equally-spaced replication is less resistant to clustering because more replicas fall into unpopulated gaps in the ring. However, at higher replication degrees, equally-spaced replication produces a relatively large number of disjoint routes and the 7-9% reduction in the number of routes can be tolerated.

Of the schemes tested, the Chord variant seems to react the best to clustering. In fact, there is a slight improvement in its performance when the number of replicas is small. In this case, the chain of replicas may span an unpopulated gap in the identifier space and replicas will be located in two clusters, which will likely produce an additional disjoint route. The performance of a few replicas spanning an unpopulated gap in the identifier space is similar to that of a chain of many replicas. This is why the benefit decreases and disappears as the replication degree increases.

As discussed, an adversary may use a run of nodes to create imbalance or disrupt the system otherwise. We measure the probability of routing success per query by determining the proportion of nodes for which a correct replica can be found along a route that contains only uncompromised nodes. The results are shown in Figure 4. Clearly, equally-spaced and random replication outperform the other replica placement schemes tested. As the theory indicates, equally-spaced replication tolerates compromised runs of length less than one-half of the identifier space well, even with a small replication degree (four replicas).

Furthermore, we investigated the possibility that an adversary may compromise a random subset of the nodes in the system. The results, shown in Figure 5, are astounding; when the adversary controls 25% of all nodes in the system, equally-spaced replication produces routes with no compromised nodes 98% of the time. Under the same level of attack, the Chord variant and spaced replication schemes successfully route only 25-60% of queries.

With only four equally-spaced replicas, we expect three disjoint routes. This implies that an adversary could prevent the success of a given query by compromising only three nodes. However, with far more nodes than that compromised in runs or randomly, almost all queries are resolved successfully. This result is extraordinary and confirms that route diversity does indeed improve routing robustness.

6. Discussion

To our knowledge, no significant work has been done to mitigate the effects of malicious routers in structured peer-to-peer systems using replica placement. We have shown that an equally-spaced replica placement scheme can improve routing robustness. To produce d disjoint routes per replica set using this scheme, each key must be replicated 2^{d-1} times. The solution requires minimal modification to the traditional Chord implementation and produces desirable results, even for sparsely populated rings. Our experiments have shown that equally-spaced replication can be used to successfully route 98% of queries in a ring that is 25% compromised.

A random replication scheme performs nearly as well as equally-spaced replication, but the complexity of the implementation makes it a less than ideal solution. Peers must apply a computationally expensive hash function or maintain a mapping of master key identifiers to replica locations

for every key. Equally-spaced replication allows peers to compute replica locations directly using addition.

Although we have discussed our work in the context of the Chord peer-to-peer system, we believe that it can be applied to any distributed hash table that routes as in [10]. Note that Chord is a prefix matching system with base equal to two. The equally-spaced replication scheme can be applied to a prefix matching system of any arbitrary base b. In this case, b^{d-1} equally-spaced replicas produce d disjoint routes. As the base increases, more and more replicas are necessary to achieve the same number of disjoint routes.

Our work has also shed light on several avenues for future work. In particular, we hope to encourage the study of replica placement as it impacts routing. We have focused on producing disjoint routes by varying the first hop in each route. However, it may be possible to produce semi-disjoint routes by varying intermediate hops within the route. One direction we have considered is the notion of *virtual rings*. The equally-spaced replication scheme can be modeled as several virtual rings (where the number of virtual rings is equal to the replication degree) implemented over a single real ring. Each virtual ring consists of the same node and key identifiers; however, the node identifiers are shifted to different real nodes. Thus, each virtual ring represents a different rotation of the original ring. However, the virtual rings need not be rotations of the real ring; they may be any arbitrary mapping of virtual nodes to real nodes. This virtual ring representation may lead to an optimal mapping that produces the greatest number of semi-disjoint routes.

In our implementation, we left some flexibility in the timing for each replica query. These queries may be dispatched in parallel, sequentially, or some combination of the two. At first glance, the dispatch of all queries in parallel might seem optimal since the correct replica with the shortest route will be returned first. However, dispatching all of the lookups in parallel may put excessive load on the system thereby degrading overall performance. The study of a real implementation or a simulation-based study of these effects would be a valuable topic for future research.

Finally, we assume that the data in the system is self-verifying. If we are able to determine the correspondence between each replica and the route along which it was found, it may be possible to select the correct object from a set of possibly incorrect replicas through a voting mechanism. This would eliminate the need for self-verifying data.

References

[1] M.S. Artigas, P.G. Lopez, A.F. Gomez Skarmeta, "A Novel Methodology for Constructing Secure Multipath Overlays," *IEEE Internet Computing*, vol. 9, no. 6, pp. 50–7, Nov. 2005.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowston, D. Wallach, "Secure Routing for Structured Peer-to-Peer Overlay Networks," *In Proc. OSDI 02*, Boston, Massachusetts, pp. 299–314, Dec. 2002.

[3] Y. Chen, R. Katz, J. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery," *In Proc. IPTPS 02*, Cambridge, Massachusetts, pp. 306–318, Mar. 2002.

[4] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Wide-area cooperative storage with CFS," *In Proc. ACM SOSP 01*, Banff, Canada, pp. 202–215, Oct. 2001.

[5] J. Douceur, "The Sybil Attack," *In Proc. IPTPS 02*, Cambridge, Massachusetts, pp. 251–260, Mar. 2002.

[6] J. Douceur, R. Wattenhofer, "Large-scale Simualtion of Replica Placement Algorithms for a Serverless Distributed File System," *In Proc. MASCOTS 01*, Cincinnati, Ohio, pp. 311–319, Aug. 2001.

[7] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, E. Zwaenepoel, "Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems," *In Proc. ACM SIGMETRICS 05*, Banff, Canada, pp. 38–49, June 2005.

[8] Q. Lian, W. Chen, Z. Zhang, "On the Impact of Replica Placement to the Reliability of Distributed Block Storage Systems," *In Proc. IEEE ICDCS 05*, Columbus, Ohio, pp. 187–196, June 2005.

[9] G. On, J. Schmitt, R. Steinmetz, "The Effectiveness of Realistic Replication Strategies on Quality of Availability for Peer-to-Peer Systems," *In Proc. IEEE P2P 03*, Linkoping, Sweden, pp. 57-64, Sep. 2003.

[10] C.G. Plaxton, R. Rajaraman, A. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment," *In Proc. ACM SPAA*, Newport, Rhode Island, pp. 311–320, June 1997.

[11] M. Portmann, S. Ardon, A. Seneviratne, "Mitigating Routing Misbehaviour of Rational Nodes in CHORD," *Symposium on Applications and the Internet 04*, Tokyo, Japan, pp. 541–545, Jan. 2004.

[12] A. Singh, M. Castro, P. Druschel, A. Rowstron, "Defending Against Eclipse Attacks on Overlay Networks," *In Proc. ACM SIGOPS 04*, Leuven, Belgium, pp. 115–120, Sep. 2004.

[13] E. Sit, R. Morris, "Security Considerations for Peer-to-Peer Distributed Hash Tables," *In Proc. IPTPS 02*, Cambridge, Massachusetts, pp. 261–9, Mar. 2002.

[14] M. Srivatsa, L. Liu, "Vulnerabilities and Security Threats in Structured Peer-to-Peer Systems: A Quantitative Analysis," *In Proc. IEEE ACSAC 04*, Tuscon, Arizona, pp.252–261 , Dec. 2004.

[15] A. Stavrou, A. Keromytis, D. Rubenstein, "Exploting Structure in DHT Overlays for DoS Protection," Columbia University Computer Science Tech. Rep. CUCS-019-04, May 2004.

[16] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *In Proc. ACM SIGCOMM 01*, San Diego, California, pp. 149–160, Aug. 2001.

[17] D. Wallach, "A Survey of Peer-to-Peer Security Issues," *International Symposium on Software Security 02*, Tokyo, Japan, pp. 42–57, Nov. 2002.

[18] M. Waldvogel, P. Hurley, D. Bauer, "Dynamic Replica Management in Distributed Hash Tables," IBM Tech. Rep. RZ-3502, July 2003.