Availability for DHT-Based Overlay Networks with Unidirectional Routing

Jan Seedorf[1] and Christian Muus[2]

[1] NEC Laboratories Europe
Kurfuerstenanlage 36, 69115 Heidelberg, Germany
jan.seedorf@nw.neclab.eu
[2] University of Hamburg
Vogt-Koelln-Strasse 30, 22527 Hamburg, Germany
christian@muus.de

Abstract. *Distributed Hash Tables (DHTs)* provide a formally defined structure for overlay networks to store and retrieve content. However, handling malicious nodes which intentionally disrupt the DHT's functionality is still a research challenge. One particular problem - which is the scope of this paper - is providing availability of the DHT's lookup service in the presence of attackers. We focus on DHTs with unidirectional routing and present concrete algorithms to extend one particular such DHT, namely Chord. Our extensions provide independent multipath routing and enable routing to replica roots despite attackers on the regular routing path. In addition, we investigate algorithms to detect adversary nodes which employ node-ID suppression attacks during routing. We demonstrate how these techniques can be combined to increase lookup success in a network under attack by deriving analytical bounds for our proposed extensions and simulating how our algorithms come close to these bounds.

1 Introduction

Distributed Hash Tables (DHTs) [12] [13] [19] [20] offer a formally defined substrate for structured overlay networks to efficiently and consistently store data items. However, in general it cannot be guaranteed that nodes in the network behave according to the DHT-protocol. This opens the door for a broad range of attacks on DHTs [2] [7] [16].

Our contribution is the enhancement of a DHT with unidirectional routing so that it can handle a high degree of adversary nodes in the network and still provide successful lookups. Unidirectional routing has the advantage that all routing paths for a particular resource converge towards the node in the network responsible for storing that resource. While this is a disadvantage from a security perspective (as we will show) this property is beneficial for caching frequently queried resources [9]. We present concrete algorithms to extend a particular DHT, namely Chord [19], while preserving an unidirectional routing structure. Furthermore, we provide a theoretical analysis of our solutions and exhibit simulation results to show the effectiveness of our algorithms.

J.A. Onieva et al. (Eds.): WISTP 2008, LNCS 5019, pp. 78–91, 2008.

In Section 2 we discuss related work and compare it to our approach. Section 3 presents a formal DHT model and our attacker model. In section 4 we define the scope of our work: lookup availability. Section 5 presents Chord, the rationale why we chose this DHT, and theoretical results on lookup availability in Chord. We present concrete algorithms for Chord extensions in section 6, including simulation results. Section 7 concludes the paper with a short summary.

2 Related Work

Much work on various DHT security challenges exists. Here we survey previous work with focus on DHT lookup availability (the scope of our work).

Srivatsa and Liu present an analytical model for the failure rate of an arbitrary lookup in DHTs [18]. They derive theoretical bounds but do not provide concrete algorithms. In a previous publication we showed that for unidirectional DHTs stronger bounds can be obtained [15].

Castro et al. investigate lookup availability in a multidirectional DHT (Pastry [13]) [2]. They suggest constrained routing tables against routing table poisoning. Further, they rely on multipath routing to derive techniques for recursive routing in a multidirectional DHT which explore alternate routing paths. In contrary, we investigate an unidirectional DHT (Chord [19]) which does not provide multipath routing. Therefore, our problem domain is different and some of our solutions are specific to unidirectional DHTs.

Danezis et al. use a weak form of a social network, the bootstrap graph, to improve lookup performance in a Chord network under attack [6]. Marti et al. use an external, existing social network to increase the lookup success rate in Chord [10]. Hence, unlike our approach, the approaches in [6] and [10] rely on the existence of a social network to increase lookup availability. However, both of these approaches are complementary to our approach and we consider using these techniques as add-ons to our algorithms interesting future research.

The approach closest to ours is Cyclone [14], an extension to Chord which can guarantee multiple independent paths in Chord in the special case where the ID-space is fully utilised. Compared to Cyclone, our solutions are beneficial in any network, independent of ID-space utilisation. In addition, our work differs from the one in [14] because we use iterative routing (which allows the detection of node-ID suppression attacks) and we directly route to replicated content for increased availability.

Contrary to our work, none of the previous extensions to Chord [6] [10] [14] considers nor mitigates the case where the node responsible for storing content (or its predecessor) is an adversary node. Not only do we consider this case in our model, additionally we provide techniques to alleviate this problem. Further, our approach is the first to enable the detection of node-ID suppression attacks on every routing hop in Chord. For our extensions to Chord we assume that secure node-ID assignment against *Sybil attacks* [7] is used. Techniques for secure node-ID assignment in a DHT have been suggested by Awerbuch and Scheideler [1], Condie et al. [3], or Fiat et al. [8] and are outside the scope of this paper.

3 Formal DHT Model and Attacker Model

The two basic primitives provided by a DHT are *store* (*key*,*data*), and *lookup* (*key*) = *data*. DHTs have been designed to guarantee consistent data storage and load balancing even when nodes enter and leave the network at a high frequency. Examples of Distributed Hash Tables are CAN [12], Pastry [13], Chord [19], and Tapestry [20]. To be able to classify threats on DHTs, precisely define the scope of our work, and to formally specify our extensions to *Chord* we use a formal model of a DHT.

3.1 Formal DHT Model

Our formal model of a DHT consists of the following:

Node-ID and Key-ID Space: An $l-bit\ key\ identifier\ space\ K$ and an $m-bit\ node\ identifier\ space\ I$ define the basic DHT structure. The DHT provides a function for mapping a key onto a key-ID k, $f_{km}(key) = k \in K$ and rules for mapping an external identifier eID onto a node identifier (*node-ID*) n_i, $f_{nm}(eID) = n_i \in I$.

Data responsibility: A data placement function $f_{dp} : K \rightarrow I$ maps a key-ID $k \in K$ onto the node-ID space I and a responsibility function $f_{resp} : I \rightarrow I$ states which node $n_i \in I$ is responsible for storing $f_{dp}(k)$. Thus, the data item for key k is stored at node $n_i = f_{resp}(f_{dp}(k))$. We denote the node responsible for storing data belonging to key k as the root node for that key $root_k$. For reliability, a replication function $f_{rep} : I \rightarrow I^r$ maps the key onto r other nodes which store the data for k as well; we call these nodes the replica roots for key k denoted by $root_k \ldots root_k^r$.

Routing: A routing table T_r at each node n_i contains t links to nodes at some distance in the ID-space. Further, a second routing table T_s at each node n_i contains s direct neighbors in the DHT structure. Routing table functions $f_{tr} : I \rightarrow I^t$ and $f_{ts} : I \rightarrow I^s$ determine which nodes are in T_r and T_s of any node n_i in the system. A routing function $f_{route} : K \rightarrow I$ specifies which entry the routing table returns upon receiving a message (lookup or storage) for key-ID k.

State: Since the system is dynamic, its state changes constantly and a set of rules for joining and leaving of nodes is necessary. As we do not examine joining and leaving of nodes in this paper we do not define these rules formally. Σ denotes the set of possible states. At any state $\sigma_i \in \Sigma$ we have N nodes in the system. The set of all N nodes (denoted $N \subseteq I$), their N routing tables T_r and T_s, and all the data items stored in the system define the current state σ_i.

3.2 Attacker Model

We assume the following attacker model: A network consisting of only good nodes is infiltrated over a certain period of time by attacker nodes which either

join the system or compromise good nodes. After this period, at a certain state $\sigma_i \in \Sigma$ the system contains $N_a = f*N$ adversary nodes and $N_g = (1-f)*N$ good nodes, where $f < 1$ and $N_a \cap N_g = \oslash$. All adversary nodes may collude (e.g., because they are controlled by a single external identity). Adversary nodes route exclusively to adversary nodes and do not drop messages: $\forall n_i \in N_a : f_{route}(k) = n_j \in N_a$ (i.e., adversary node *suppress* existing good nodes in their routing tables); good nodes route to good and adversary nodes: $\forall n_i \in N_g : f_{route}(k) = n_j \in N$.

In principle, adversary nodes could also drop messages. However, this would result in a less severe attack on lookup availability because this behaviour can easily be detected through time-outs. In contrary, by continuing to route amongst them (never reaching the target data item) colluding adversary nodes can absorb more DHT routing resources in vain. Thus, by expecting adversary nodes not to drop messages we consider a stronger attacker model.

Adversary nodes are distributed uniformly[1] over the node-ID space I. Additionally, we assume that any message sent on a single DHT-hop will arrive unchanged (i.e., attacks on the IP-layer are out of scope).

4 Availability of the Lookup Service

In principle, without a trusted authority in the network, a single adversary can control a large fraction of an overlay network with only a few external identities [7]. An adversary node on the path from the query node to some key can either drop the message, alter the message, or route the message to another adversary node. Castro et al. were the first to thoroughly investigate this problem [2]. They conclude that in order to achieve *secure routing* in a DHT three properties have to be fulfilled: 1) *Secure node-ID assignment*, 2) *Protection against routing table poisoning*, and 3) *Secure message forwarding*. In this context we define *lookup availability* as follows:

Definition 1 *The* Availability of the Lookup Service *is the probability that the corresponding data item is returned by the DHT after a node has invoked an arbitrary lookup for a key.*

A lookup can consist of many routing attempts from the query node to the key. Thus, a lookup can use several different paths and is finished if either it succeeds, a threshold t_h (limiting the number of hops used in the lookup) is reached, or all possible paths between the query node and the node responsible for storing the corresponding data item (i.e., $root_k$) have been tried without success. We define a path in a DHT as follows:

[1] Existing work on secure node-ID assignment for DHTs and for Chord in particular [1] [2] [3] [8] provides solutions to achieve this property. Thus, this assumption is reasonable if secure node-ID assignment techniques are used. We expect the use of such techniques as a fundament for our extensions (see further section 4).

Definition 2 *A* path $p(n_q, k) \subseteq N$ *from a query node* $n_q \in N$ *for key* $k \in K$ *is any set of nodes such that routing from* n_q *for key* k *will pass through these nodes including* $root_k$*. Two different paths are called* alternate *if at least one node (other than* n_q *and* $root_k$*) is on both these paths and* independent *if they share no common node other than* n_q *and* $root_k$*.*

As a metric for lookup availability in a DHT we use the success-rate of a random lookup (as a secondary metric we use the hop count of a random lookup, denoted with χ):

Definition 3 *The* success rate ρ *is the probability that a random lookup will succeed:* $\rho = P(\exists p(n_q, k) | \forall n_i \in p(n_q, k) : n_i$ *is good*$)$ *where* n_q *is a random query node and* k *is a random key.*

We assume that secure Node-ID assignment techniques against *Sybil attacks* [7] are used [1] [2] [3] [8] and that the DHT is protected against routing table poisoning (*Eclipse attacks* [16]): $f_{nm}()$, $f_{tr}()$, and $f_{ts}()$ cannot be attacked. This implies that at state σ_i in a reasonably large network the routing table T_r of any good node in the system contains with high probability $f \times d$ adversary nodes and $(1 - f) \times d$ good nodes, where $d \leq t$ is the number of distinctive nodes in T_r. Further, we assume that the integrity of data items stored in the DHT can be verified by the application on top of the DHT, e.g., by using a public key infrastructure or self-certifying keys/data [5].

Despite these assumptions attackers are still able to degrade the availability of the DHT severely by attacking the routing function $f_{route}()$, i.e., message forwarding. Our goal is to develop algorithms for $f_{route}()$ that provide resilience against such attacks on the DHT-layer.

5 Extending an Existing Unidirectional DHT

As an example DHT with unidirectional routing we choose Chord [19]. Our goal is to make as few general changes to regular Chord as necessary. In fact, we only make very few changes to Chord that have to be adopted by all nodes in the system (which we call *global* extensions). These changes do not change Chord's formal properties. Most extensions we introduce are *local*: nodes can optionally decide to use a different $f_{route}()$ function than in regular Chord. However, these local extensions do not affect other nodes or the DHT.

Chord uses the IP-address of a node as its external identifier (eID). A predefined hash function $h()$ maps any eID onto an $m - bit$ node-ID n_i and also any key onto an $m - bit$ key-ID k. The node identifier space I is a virtual ring where node-IDs are ordered clockwise from 0 to $2^m - 1$. Each node in the ring is responsible for storing the content of all key-IDs that are equal to or less than its own identifier but larger than the identifier of the node's direct predecessor in the Chord ring. For reliability against node failures, the data for k is also stored at r nodes directly succeeding $root_k$ in the ring. In its routing table T_r each

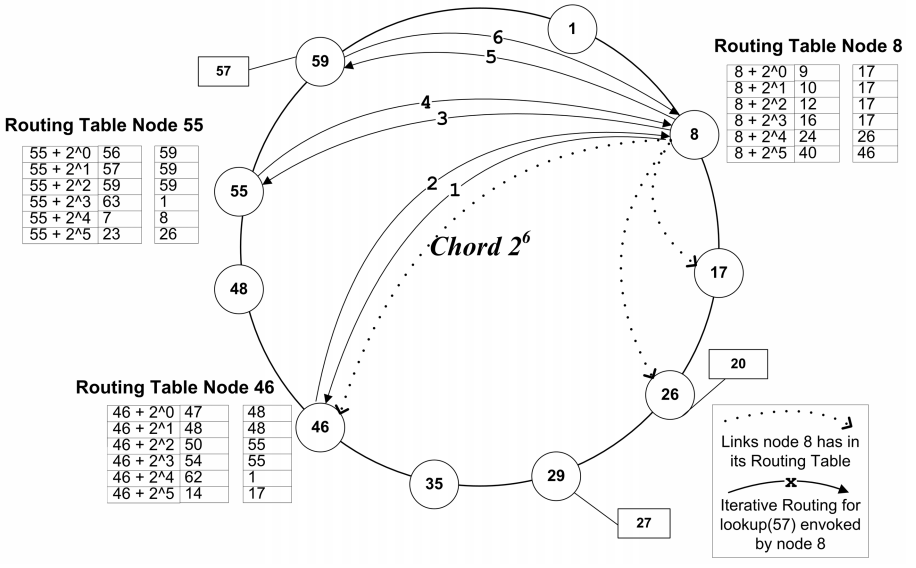

Fig. 1. Iterative Routing in Chord

node n_i stores links to m succeeding nodes in the ring (*unidirectional routing* [9]). Additionally, each node keeps a link to its direct predecessor in the ring.

It is precisely specified how routing tables are filled (making routing tables *constrained*, protecting against *Eclipse attacks* [2] [16] and thus making our assumption of $f_{tr}()$ and $f_{ts}()$ being secure reasonable for Chord): The jth entry in T_r contains the IP-address of the first node that follows n_i by at least 2^{j-1} in the virtual ring: $f_{tr}(n_i) = [succ(n_i + 2^0), succ(n_i + 2^1), ..., succ(n_i + 2^{m-1})]$ where $succ(x) = n_o \geq x(\neg\exists n_p \in N | n_o > n_p \geq x)$. '>' is a relation 'succeeding in the ring' using modular arithmetic to ensure that routing and data responsibility is shared across $2^m - 1$ in the ring. The first entry in T_r is the node directly succeeding n_i. The last entry in T_r contains a link to a node at least $\frac{2^m}{2}$ away from n_i in the ring. To achieve fast lookups, nodes forward messages to the node with the highest ID in their routing table that is smaller than the key-ID (greedy routing). Routing succeeds when the direct successor of a node has a larger ID than the key-ID. This successor node is responsible for the key. Additionally, each node keeps a list of its s direct successors T_s to handle node failures. Routing is either iterative (the query node contacts other nodes to get iteratively closer to the key) or recursive (a query message is passed through the network hop by hop).

Figure 1 exemplifies iterative routing in a Chord network [15]. In the routing tables displayed the rightmost column shows to which other nodes in the DHT links exist. The two leftmost columns point out how to compute precisely which node is in the particular routing table entry, i.e., determining the value where the first node 'succeeding' this value in the ring must be in that routing table entry (compare to the previous paragraph). In this example, a query node with

node-ID 8, n_8, starts a lookup for key-ID 57. n_8 sends a message to the node in its routing table that has the node-ID closest to but not larger than the key-ID (57) which is n_{46} in this example $\langle 1 \rangle$. n_{46} replies by returning to n_8 the node with the highest node-ID from its routing table not larger than the key-ID which is n_{55} in this example $\langle 2 \rangle$. n_8 sends out a query message to n_{55} $\langle 3 \rangle$. n_{55} determines that the first node in its routing table, i.e., its direct successor in the ring, has a node-ID (59) which is higher than the key-ID (57). n_{55} concludes that this node must be responsible for key-ID 57 and returns n_{59} to the query node n_8 $\langle 4 \rangle$. To retrieve the data item for key-ID 57, n_8 contacts n_{59} $\langle 5 \rangle$ which answers by sending the corresponding data item to n_8 $\langle 6 \rangle$.

In regular Chord, any lookup has to pass the predecessor of the node storing the content for the key looked up. This is also referred to as the *shield problem* [11] [15] and a consequence of unidirectional greedy routing. We denote the predecessor of $root_k$ with $shield_k$ for any key k. Formally, we define $shield_k = n_i \in N | (n_i < root_k) \wedge (\neg \exists n_l \in N | n_i < n_l < root_k)$.

An important consequence of the shield problem is that in Chord only one independent path from the query node n_q to $root_k$ exists for any lookup. Hence, the success rate for an arbitrary lookup in regular Chord is bound by the following inequality [15]:

$$P(lookupsuccess) \leq (1-f)^2 \tag{1}$$

To see why inequality (1) holds, consider a random lookup for a key. In our model, with probability $(1-f)$, $root_k$ is good and with probability $(1-f)$, $shield_k$ is good. Any lookup can only succeed if both nodes are good because any lookup has to pass these two specific nodes. Since it is statistically independent if either one of the two nodes is controlled by an adversary equation (1) holds.

In [18] an upper bound for DHTs is given on the failure rate for an arbitrary lookup which can be converted into a lower bound on the success rate by taking the opposite event and mapped to Chord [15]:

$$1 - \left(1 - (1-f)^{\left(\frac{1}{2}\right)\log N}\right) \leq P(lookupsuccess) \tag{2}$$

6 Algorithms for Increased Lookup Availability

In this section we describe our extensions to Chord for increasing lookup availability. In principle, we combine three techniques: 1) We use the direct successor list of each node to accomplish independent multipath routing. 2) To overcome the shield problem we directly route to replica roots. 3) We use density checks on each iterative routing hop to detect paths that contain adversary nodes as early as possible.

For all our techniques described below we use the following general (global) extension to Chord: Each node in the network must support iterative routing where at each routing hop the query node receives not only the next hop from the node it queried (as in regular Chord) but instead the whole routing table T_r of

the queried node and its list of direct successors T_s. Note that this extension only affects the size of each iterative query response. In particular, it does not affect the total number of links stored at each node because the additional information received at each iterative routing step is only stored temporarily during the lookup. We call this extension *complete-knowledge iterative routing* because at each iterative routing hop the query node receives the complete information the hop node has about the network. All other routing techniques we introduce are solely computed at the query node (locally). Thus, it does not affect the success rate of a lookup if other nodes in the network use these techniques or not.

6.1 Multiple Independent Paths

In the case a lookup path has failed, we explore two techniques to let the lookup continue (we refer to this as *failover routing*)[2]: a) by starting a new independent path at the query node (*independent restart*) or b) by starting a new path at the closest node to the key received during the previous path which has not been used in the lookup (*backtracking*).

For both techniques the query node maintains a temporary list T_m of nodes it has used in the lookup so far. In each individual path it explores during a lookup the query node only uses nodes it has not used before in this lookup, i.e., nodes $\notin T_m$. In regular Chord the direct successor list T_s is only used for redundancy (i.e., in the case of node failures). We allow each node to use the list of direct successors T_s on every routing hop. Since we use complete-knowledge iterative routing a query node can in principle use for the next hop any node from the routing table T_r and the direct successor list T_s it received from the node on the last hop. However, for our extensions at each hop the nodes in T_s are only used in routing if all nodes from T_r have been used previously in the lookup, i.e., are already in T_m. n_q (the query node) always routes greedy (as in regular Chord): It always uses the node $n_i \in T_r$ (or $n_i \in T_s$ if $\forall n_i \in T_r : n_i \in T_m$) with the highest node-ID smaller than k. This assures that queries make progress.

With unidirectional greedy routing independent paths converge towards the root [9]. Using T_s allows a path to continue if at some hop in T_r all entries smaller than k are already in T_m. For independent restart, using T_m guarantees that all paths in a lookup are independent. Further, independent restart allows for up to s (the number of entries in every T_s) independent paths because this is the maximum number of independent paths that can converge on the penultimate hop before reaching the root. Because with backtracking a new path does not start at the query node, this technique explores alternate (not independent) paths.

In our model, adversary nodes suppress good nodes in the routing tables T_r and T_s they return. This implies that once a path has reached an adversary node, only adversary nodes will be added to T_m on this path. Thus, node-ID suppression attacks do not prevent our technique to subsequently explore a path with only good nodes on every hop.

[2] Remember that in our model a lookup consists of several individual paths.

6.2 Direct Replica Routing

To tackle the situation where $root_k$, $shield_k$, or both are malicious we allow to route directly to the replica roots of k. Chord replicates content at r replica roots which are the r nodes directly succeeding $root_k$ in the ring. However, in regular Chord the replica roots are only used for redundancy (i.e., node failure of $root_k$).

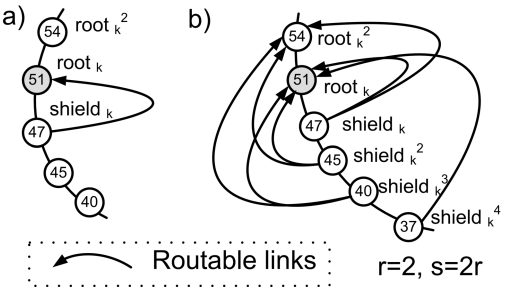

Fig. 2. Direct Replica Routing

We now extend Chord in a way that routing to the replica roots of a key k is possible without passing $shield_k$ nor $root_k$: We allow *direct* routing to a node $n_i \in root_k \ldots root_k^r = REP_k$ if $n_i \in T_s$ (we refer to this as *direct replica routing*). Because at every hop T_s contains s direct successors in the ring, the query node can check if some of these nodes are $\in REP_k$ (n_q simply has to verify if $\exists n_j \in T_s | k \leq n_j \leq root_k^r$). If all replica roots retrieved at some hop have been queried without success, a failover (backtracking or restart) is pursued.

Using direct replica routing results in each key k having effectively s shield nodes (the s direct predecessors of $root_k$) which we denote with $shield_k ... shield_k^s = SHI_k$. By setting $s = 2r$ (globally) in the system, any of the $r + 1$ closest shield nodes to a particular key k can route directly to any of the r replica root nodes for k. In general, setting $s \geq r$ ensures that the last replica root $root_k^r$ is accessible from $s - r + 1$ shield nodes.

Figure 2 exemplifies how replica roots can be reached through more than one node (b) compared to regular Chord (a). Any T_s the query node n_q will receive from an adversary node will only contain the next s adversary nodes in the ring. However, by setting $s = 2r$ we guarantee that reaching one good shield node of the r closest shield nodes to k is enough to reach one good replica root $\in REP_k$ (if existing).

6.3 Detecting Node-ID Suppression Attacks

Recall that in our attacker model a network of good nodes is infiltrated and routing tables in Chord are constrained (and therefore protected against routing table poisoning). Thus, good nodes have (with high probability) $f \times d$ adversary nodes and $(1-f) \times d$ good nodes in their routing table T_r as well as $f \times s$ adversary nodes and $(1-f) \times s$ good nodes in T_s. Adversary nodes suppress good nodes in the routing tables they return. This enables them to attack lookup availability even if complete-knowledge iterative routing is used by the query node.

We can detect these attacks by using *density checks*: the query node n_q calculates the average distance α between nodes in its direct successor list T_s as

$\alpha = \frac{n_l - n_f}{s}$ where n_l is the last entry in T_s and n_f is the first entry in T_s. From any routing table $T_s(n_i)$ that n_q receives from a node n_i, n_q can compute $\alpha(n_i)$ and compare it with its own average distance by computing $\delta = \frac{\alpha(n_i)}{\alpha(n_q)}$. If $\delta \geq t_d$ (a density threshold), n_q considers n_i to be an adversary node.

An adversary node n_a can only decrease its $\alpha(n_a)$ by either creating artificial entries in $T_s(n_a)$ (which will be detected on the next hop if such an entry is chosen by n_q) or limit suppression of good nodes in $T_s(n_a)$ (which would give n_q access to good nodes). With a low density threshold t_d there is a risk of falsely estimating good nodes as adversary ones. However, this only affects $f_{route}()$ of n_q locally.

6.4 Theoretical Analysis of the Proposed Extensions

Our proposed extensions to Chord provide several independent paths between n_q and $root_k$, and route directly to the replica roots of a key k so that not a single root node can control all access to data items for a key k. Thus, there exist at most s shield nodes (one on the penultimate hop of every independent path) denoted $shield_k \ldots shield_k^s$ and for every key k there are r routable replica roots, denoted $root_k \ldots root_k^r$.

We now extend the theoretical results for regular Chord from Section 5 to this case. Analytically, we use a sample space Ω for a random lookup. Ω samples all shields and all replica roots for an arbitrary key and determines for each shield and replica root node if it is an adversary node. We are interested in the following events in our sample space:

$$A = \{\textit{"at least one shield node is good"}\} \quad B = \{\textit{"at least one replica root is good"}\}$$
$$E = \{\textit{"at least one replica root and one shield node are not adversary nodes"}\}$$

Event E states an upper bound on the success rate for an arbitrary lookup because this event is the minimum requirement for any lookup to succeed (a lookup can still fail under this event if all paths explored contain at least one adversary node).

We now derive the probability for event E for the case that we have precisely s shield nodes and r routable replica roots for any key k:

$$P(A) = 1 - f^s \quad P(B) = 1 - f^r \tag{3}$$

$$P(lookupsuccess) \leq P(E) = P(A) * P(B) = (1 - f^s) * (1 - f^r) \tag{4}$$

Note that it is possible to multiply P(A) and P(B) because these events are statistically independent in our model. Adapting the lower bound from inequality (2) to s independent paths we get [18]

$$1 - \left(1 - (1 - f)^{\left(\frac{1}{2}\right)\log N}\right)^s \leq P(lookupsuccess) \tag{5}$$

With our extensions, there exist at most s independent paths and exactly r replica roots. Since equation (4) provides an upper bound, it holds for our extensions even though some lookups might explore less than s independent paths.

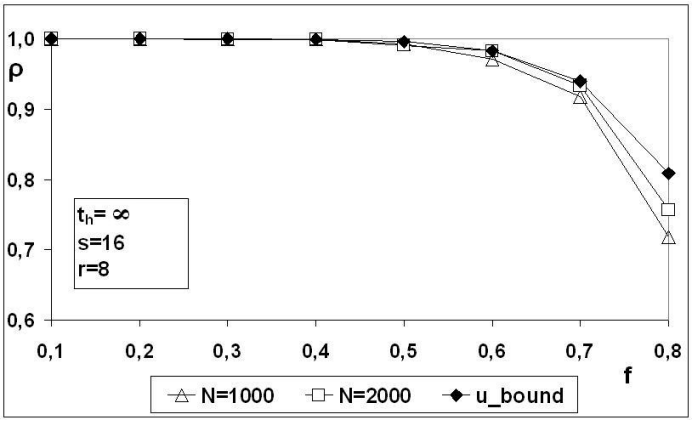

Fig. 3. $MRR - r$ with $t_h = \infty$ compared to theoretical upper bound (N= 1000/2000)

However, the lower bound in inequality (5) does not apply to our extensions. Still, it indicates analytically that as more independent paths are explored (which is the effect of our multipath-extensions to Chord) the lower bound on the success rate increases. In any case, we are interested in the maximum success rate (and thus the upper bound) that our extensions can theoretically achieve.

6.5 Simulations

To see how close our algorithms come to theoretical limits we simulated multipath routing combined with direct replica routing (which we call *MRR* for *Multipath Replica Routing*) for various network sizes N and attacker rates f and compared it to the upper bound on ρ from equation (4). In all our experiments we simulated 1000 lookups in 10 random Chord networks with $|I| = 2^{32}$ and adversary nodes behaving according to our attacker model. We only consider lookups where $T_s(n_q) \cap REP_k = \emptyset$, i.e., lookups where no replica root is contained in the direct successor list of the query node.

Figure 3 shows results for independent restart (we also conducted simulations for backtracking with very similar results). It can be observed that our algorithms come very close to the upper bound (*u_bound*) on lookup success in equation (4), almost reaching theoretical limits even for high attacker rates.

We noticed however that with $t_h = \infty$ (as in Figure 3) the average hop count χ can get quite high with increasing levels of network infiltration (e.g., for $f = 0.7$, N= 2000 and a success rate of 92% we obtained an average of 635 hops per lookup). In some applications for which DHTs have been proposed (e.g., signalling in real-time communications [17] or a distributed DNS architecture [4]) the time it takes for a lookup to succeed is crucial. To reflect this requirement and investigate the effectiveness of our algorithms with a timing constraint, we conducted simulations with a hop threshold t_h. Figure 4 displays MRR with backtracking (-b) and independent restart (-r) compared to regular Chord with

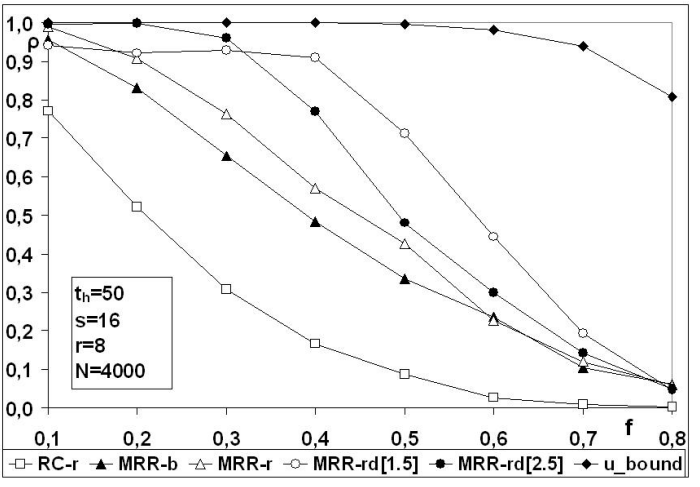

Fig. 4. Success rate for MRR compared to regular Chord and upper bound ($t_h = 50$, N= 4000)

independent restart (RC-r) for $t_h = 50$. Additionally, the figure shows the success rate for MRR-r with density checks (MRR-rd) for $t_d = [1.5, 2.5]$.

It can be noticed that independent restart performs better than backtracking for attacker rates up to $f = 0.6$. Further, the detection of node-ID suppression attacks with density checks on every hop increases lookup availability perceptibly. One can see that a higher threshold t_d is better suited for low attacker rates whereas a lower threshold results in better performance for high attacker rates (in Figure(4), for attacker rates higher than $f = 0.3$, $t_d = 1.5$ performs better). In general, it is advisable to set $t_d < \frac{1}{f}$ because the range of an attacker's successor list increases reciprocal to f with node-ID suppression attacks.

In addition to increasing the success rate, density checks also significantly decrease the hop count χ. Table 1 illustrates this by showing ρ and χ for MRR-r (with and without density checks) with f=0.6 and N=2000. Compared to MRR-r without any hop threshold, density checks achieve a more than 35 % lower success rate. However, note that the average hop count needed for this result is a factor of 5 lower. Compared to MRR-r using the same hop threshold (t_h = 100), routing with density checks requires ∼14/6 less hops on average (t_d = 1.5/2.5) even though it achieves a higher success rate. We consider exploring the tradeoff between ρ, χ and t_h interesting future research (in the end, deciding on this tradeoff depends on application constraints/demands).

Table 1. ρ and χ for MRR-r (f=0.6, N=2000)

	t_h	t_d	ρ	χ
MRR-r	100	∞	0.49	74.1
MRR-r	∞	∞	0.98	321
MRR-rd	100	1.5	0.62	59.8
MRR-rd	100	2.5	0.61	68.1

7 Conclusion

We enhanced a DHT with unidirectional routing (Chord) to increase lookup availability. Our proposed algorithms enhance Chord with independent multipath routing, direct routing to replica roots, and mechanisms for detecting node-ID suppression attacks to provide resilience of the DHT's lookup service against attacks on the DHT-routing layer. We showed through simulations that our algorithms can come very close to theoretical limits. For example, we can achieve a lookup success rate of 98 % in a network with 60 % adversary nodes.

We consider combining our algorithms with techniques relying on social networks on top of a DHT (see related work) as well as exploring the tradeoff between the hop threshold, the average hop count, and the success rate interesting future research.

Acknowledgement

The authors would like to thank Rolf Winter for providing useful suggestions to an earlier version of this paper and the anonymous reviewers for providing valuable comments which improved the quality of the paper.

References

1. Awerbuch, B., Scheideler, C.: Towards Scalable and Robust Overlay Networks. In: Sixth International Workshop on Peer-to-Peer Systems, IPTPS 2007, Bellevue, WA, USA, February 26-27 (2007)
2. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. In: Proc. of the 5th Symposium on Operating Systems Design and Implementation, Boston, MA, December 2002, ACM Press, New York (2002)
3. Condie, T., Kacholia, V., Sankararaman, S., Maniatis, P., Hellerstein, J.M.: Maelstrom: Churn as Shelter, University of California at Berkeley Technical Report No. UCB/EECS-2005-11 (November 2005)
4. Cox, R., Muthitacharoen, A., Morris, R.: Serving DNS Using a Peer-to-Peer Lookup Service. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
5. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Proc. of SOSP 2001, Banff, Canada (2001)
6. Danezis, G., Lesniewski-Laas, C., Kaashoek, M.F., Anderson, R.: Sybil-Resistant DHT Routing. In: de Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 305–318. Springer, Heidelberg (2005)
7. Douceur, J.R.: The Sybil Attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
8. Fiat, A., Saia, J., Young, M.: Making Chord Robust to Byzantine Attacks. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 803–814. Springer, Heidelberg (2005)
9. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials 7(2), 72–93 (2005)

10. Marti, S., Ganesan, P., Garcia-Molina, H.: DHT Routing Using Social Links. 3rd Int. Workshop on Peer-to-Peer Systems (2004)
11. Muus, C.: Availability in DHT-based Structured Overlay Networks Considering Chord as an Example, Diploma Thesis, University of Hamburg, Germany (November 2007)
12. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. of SIGCOMM 2001, San Diego, USA, August 27-31 (2001)
13. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany (November 2001)
14. Sanchez Artigas, M., Lopez, P.G., Skarmeta, A.F.G.: A Novel Methodology for Constructing Secure Multipath Overlays. IEEE Internet Computing 9(6), 50–57 (2005)
15. Seedorf, J., Muus, C.: Availability for Structured Overlay Networks: Considerations for Simulation and a new Bound on Lookup Success. In: 12th Nordic Workshop on Secure IT-Systems - NordSec 2007, Reykjavik, Iceland (October 2007)
16. Singh, A., Castro, M., Druschel, P., Rowstron, A.: Defending against eclipse attacks on overlay networks. In: Proc. of the ACM SIGOPS European Workshop (September 2004)
17. Singh, K., Schulzrinne, H.: Peer-to-Peer Internet Telephony using SIP. In: Proc. of the international workshop on Network and operating systems support for digital audio and video, Stevenson, Washington, USA, June 2005, pp. 63–68. ACM Press, New York (2005)
18. Srivatsa, M., Liu, L.: Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis. In: Proc. of the 20th Annual Computer Security Applications Conference (ACSAC), Tucson, Arizona, December 6-10, 2004, pp. 251–261. IEEE CS Press, Los Alamitos (2004)
19. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. In: IEEE/ACM Transactions on Networking, February 2003, vol. 11(1), IEEE Press, Los Alamitos (2003)
20. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 22(1) (January 2004)