Man-in-the-Middle Attacks in Distributed Hash-Tables

Thomas Reidemeister, Klemens Böhm, Erik Buchmann, Paul A.S. Ward

Abstract

Distributed hash-tables (DHTs) are peer-to-peer overlay networks that are primarily used for distributed data storage. While there are several design variants of DHTs, including the Content-Addressable Network (CAN), Chord, and Pastry, all promise to be self-organizing, fault-resistant, and highly scalable. However, most of the proposed protocols do not deal with the threat of man-in-the-middle-attacks, and insofar as they attempt to address it, the security provided is extremely limited.

In this paper we analyze of the threat of man-in-the-middle (MITM) attacks on such protocols, with a detailed focus on such attacks in the CAN protocol. We demonstrate that they are far more damaging than end-node attacks. Specifically, the harm of end-node attacks is equal to the number of nodes corrupted. By contrast, damage caused by MITM attacks grows proportional to the number of nodes corrupted and exponentially with the size of the network. We provide an overview of different techniques that reduce the impact of such attacks at moderate cost, with a focus on three replication-based approaches. We analyzed the effectiveness of our countermeasures, providing a model in order to estimate the attack probabilities where possible. We validate our formal analysis by simulation over a large CAN.

Index Terms

peer-to-peer overlay, distributed hash-table, man-in-the-middle attack, malicious behavior, networking, distributed systems

I. Introduction

Distributed Hash Tables (DHTs) are designed to manage huge quantities of (key,value)-pairs and perform large numbers of parallel operations of the form `add(key,value)`,`delete(key)`, `update(key,value)`, and `value=lookup(key)` on a common key space. The DHT consists of many anonymous and autonomous peers that organize as an overlay network on top of large physical networks. Each peer is responsible for a certain fragment (zone) of the key space. Operations are addressed with a key in the key space. The peers forward any operation to an adjacent peer whose zone is closer to the key of the operation than their own zone, until the operation reaches the peer that is responsible for the addressed zone. Prominent examples of DHTs include the Content-Addressable Network (CAN) [1], Chord [2], and Pastry [3]. They differ primarily in the structure of the key space, routing path selection, and contact selection.

Current and proposed applications of DHTs include distributed backup systems [4], massively distributed query systems like PIER [5], and publish-subscribe frameworks [6]. While some uses are anticipated to occur in closed environments, typically behind corporate firewalls, many usages envisioned will, either of necessity or desire, be open in nature. For example, Huebsch *et al.* [5] propose using a DHT-based database for intrusion detection in TCP/IP networks. In their approach, the users generate fingerprints for each recognized attack and store them in a Content-Addressable Network. Fingerprints of suspicious occurrences are compared with fingerprints of prior attacks recorded in the CAN in order to detect intrusions. Such an application must, by its very nature, operate in an open DHT.

Any DHT that operates in such an open environment will be subject to misuse. Vulnerabilities exist below the DHT level (*i.e.*, at transport layer or below), at the DHT protocol level itself, and at the application level. This paper focuses on security at the DHT protocol level.

In general, misuse of the DHT protocol can affect the consistency, reliability, and performance of any application that is built on top of that DHT. There are three broad forms of misuse: free riders, errors, and deliberate attack. By free riders we mean lazy peers. Such peers do not actively attack the system, but will not aid it unless given an incentive. They wish to gain the benefits of the system, without incurring any cost. Various countermeasures against such free riders have been developed [7], [8] and appear to be effective. Errors in any implementation of the DHT protocol may cause a peer to not perform according to the DHT protocol specification. The form of failure may be anything from fail-silent (*e.g.*, dropping operations rather than forwarding them) to Byzantine. A Byzantine failure cannot be formally distinguished from a deliberate attack. Deliberate attack is the act of *malicious peers* intentionally harming the correct operation of the DHT protocol. Such attacks, at the DHT protocol level, occur either on route to the destination, or at the end-node. Attacks that occur on route are referred to as man-in-the-middle (MITM) attacks. They can disrupt the communication, either dropping or modifying the operation, or masquerade as the end-node.

Measures to preclude free riders do not appear to be effective in dealing with either Byzantine failure or deliberate attack on the DHT protocol, and there are no other proposed countermeasures that we are aware of. In particular, malicious peers can collaborate (specifically, a single attacker could operate a large number of malicious peers running on machines that were infected by Trojans [9]), or a single peer could register under various identities [10]. Single peers with multiple identities can so try to misuse the maintenance algorithm to populate the neighborhood of a number of peers and disrupt their routes [11]. Collaborating malicious peers violates the assumptions required for countermeasures to free riders. As such, this paper focuses on the problem of securing DHT protocols in general, and the CAN protocol specifically, to such deliberate attacks.

In prior work [12], we have shown by simulation that in a CAN consisting of 10,000 peers only 1% malicious peers are necessary to attack 8% of the operations. In this paper we will show that the probability of a MITM attack is exponential in the number of peers and linear in the number of attackers, where the constant is equal to the average path length. By contrast, attacks purely on the end nodes are directly equal to the number of attackers. As such, the major contribution of this work is to present extensions to the CAN protocol to address the problem of MITM attacks. We analyze the effectiveness of our approach, and validate our results with simulations over a large CAN.

The remainder of the paper is organized as follows. In Section II we discuss the existing literature on MITM attacks, and review its applicability to our specific DHT problem. We then provide the structure and routing algorithm used in the CAN protocol, as necessary to understand this work. In Section IV we describe the different characteristics of MITM attacks, and show the extent of damage a given percentage of malicious peers can cause. We provide an overview of our countermeasures in Section V, describing three replication-based approaches in detail. After formally analyzing the methods, we present a summary of our detailed simulation studies of the protocol extensions in Section VII. We conclude by discussing the trade-offs in the approaches, as revealed in the analysis and simulation and providing a short survey of related work.

II. Related Work

In this Section we provide a general overview of countermeasures that are applicable to counter

MITM attacks, or can be adapted to do so. The countermeasures are reviewed according to their applicability in DHTs. There are two basic approaches that can be taken: message checking and replication.

The message-checking approach involves sending a single operation to the destination peer and attempts to verify that the desired peer is reached with an uncorrupted message. There are three approaches that can be used for this verification process. Sit and Morris [13] proposed the first method. Given that routing in DHTs follows a deterministic routing protocol (*Greedy Forwarding*, *cf.* Section III), where the distance of a message to the addressed peer decreases with each forward, they propose a distributed algorithm to observe routing paths in DHTs. According to their proposal, a straightforward implementation would work as follows: (1) Each forwarder sends a confirmation to the sender of the message. This confirmation contains the message id, the coordinates of the local zone, and the contact information of the current forwarder. (2) The sender monitors the routing path of the message and checks its validity, *e.g.*, by calculating the distance from each confirmed zone to the destination key. (3) If the message got lost, the sender uses the last confirmation to order a retransmission of the message to another peer. However, this approach does not prevent the payload of the message from being modified. Furthermore, a retransmission increases the latency of the operation, and sending confirmations is also likely to increase the bandwidth overhead. The peer is also in a guessing situation which peer has modified the message. This approach is not resistant against colluding peers. A path consisting of a sequence of malicious peers could send spoof confirmations to hide an attack.

The second verification technique assumes there is a public-key infrastructure (PKI) [14]. The sender signs each message. The destination peer obtains the public-key from a trusted authority and verifies the authenticity and integrity of the message. However, using a central authority does not conform to the concept of peer-to-peer networking. Furthermore, using a PKI to avoid masquerading would require *a priori* knowledge about the designated destination peer (*i.e.*, its public-key) of a message that is usually not known in advance.

Finally, rather than verify individual messages, the concepts of trust and reputation can be employed. For example, Aberer and Despotovic [8] propose the use of a fully featured "parallel" DHT that stores complaints of the peers. A peer files a complaint about another one if it obtains a query result that it deems unsatisfactory, and uses a complex metric to derive a trust value from the number of complaints about a certain peer. The major problem with this approach is that it begs the question: how do we know that the parallel DHT is not the victim of MITM (or other) attack?

The replication method sends the operation over multiple (hopefully disjoint) paths with the presumption that it is doubtful that a MITM attack will affect all paths. In order to improve not only fault tolerance but also data availability and path latency, Ratnasamy *et al.* [1] propose to use multiple independent key spaces. They call this concept *multiple realities*. Each peer manages a different zone in each reality (key space). In order to query a certain key, a message is sent to the neighbor in the reality that has the shortest distance to the destination key. If a routing failure occurs, a message is sent in another reality. Multiple realities provide an efficient way to deal with lost messages and faulty peers. However, the approach is not designed to deal with MITM *per se*. In Section V we will extend this concept to help preclude MITM attacks.

Finally, Castro *et al.* [15] propose the use of redundant routing to counter MITM. They developed a routing algorithm for Pastry that reduces the impact of MITM attacks. If an attack is detected, messages are sent on different routes to the same destination. However, to avoid malicious peers from taking over a large part of the network, all peers are required either to pay fees to get a key from a trusted authority or to solve cryptography puzzles. This procedure reduces the issue of single malicious peers joining multiple times into the network by exposing them to unnecessary overhead. The overhead may hinder peers with limited computing power or limited bandwidth to even join the overlay.

III. The Content-Addressable Network

The Content-Addressable Network (CAN) is a prominent realization of a DHT overlay. Its key space is organized as a multi-dimensional torus. That is, the key space wraps in each dimension from 0 to 1 and vice versa. Each peer is responsible for a certain partition of the key space. In the following, we denote the dimensionality of the key space as d and refer to each partition of the CAN as a *zone*.

Each peer maintains a set of neighbors, storing both contact information and the dimensions of the zones of adjacent peers. We define two peers as *neighbors* if their zones have one point in common.[1] The peers are able to perform the operations `add(key,value)`, `value=lookup(key)` and `delete(key)`. Thus, each operation is addressed with a key. The *keys* are Cartesian coordinates $(x_1, x_2, \ldots, x_d)$ mapped onto the key space. If a peer cannot perform an operation on its local zone (*i.e.*, the key for the operation does not reside within the dimensions maintained by the peer), it forwards the request to that neighbor whose zone is closest to the key of the operation. The distance between the message key and the zone of a peer is the Euclidean distance between their coordinates in the key space. We refer to this routing as *single path routing*.

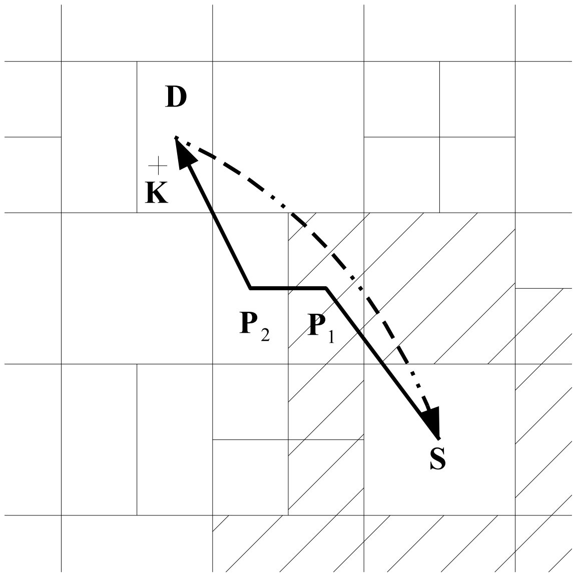


Fig. 1. Routing in a 2-Dimensional CAN

Figure 1 shows an example of a two-dimensional key space. The zones are shown as rectangular areas. Suppose sender S issues a request `lookup(K)` for key K which is contained in the zone of peer D. Peer S now creates a *request message* containing the destination key K, the contact information of the sender S, and additional data of the operation to be performed. The peers shown with hatched zones are the neighbors of peer S. In order to route the request according to greedy forwarding, peer S computes the distance from the zone of each of its neighbors to key K. In our example, peer P_1 is neighbor of peer S that is responsible for the zone which is closest to key K. Thus, peer S forwards the request message to peer P_1.

[1]This definition differs from the original proposal of the CAN, where two peers are neighbors if their zones overlap in $d-1$ dimensions. We do this because our protocols require a large number of neighbors to enable disjoint paths between the peers. Increasing the number of dimensions in order to have a large number of neighbors generates a larger maintenance overhead than simply changing the policy for contact selection.

Peer P_1 will then perform the same algorithm, forwarding the message to peer P_2, which in turn forwards it to peer D. Peer D performs the requested operation and sends the result directly back to peer S, since the necessary contact information is attached to the request.

We refer to peers issuing the original operation, in this case peer S, as *source peers* and to peers that manage a zone containing the key, in this case peer D, as *destination peers*. Peers that forward the requests without performing, in this case peers P_1 and P_2, are termed *forwarders*. The sequence of forwarders forms a *path* from the source to the destination.

IV. Man-in-the-Middle Attacks

Misuse of the DHT protocol may occur for a number of reasons, though we can identify three broad categories:

Cost As with any P2P network consisting of rational participants, the peers in DHTs prefer to reduce their workload without taking care of others. Peers could drop requests or route them away from the own zones in the direction of other peers. Such peers are known as free riders.

Error Technical problems can result in messages that are modified while being forwarded or answered. Faulty peers can corrupt or drop messages due to poor protocol implementations, or other system failures.

Malice A malicious peer can try to prevent other peers from inserting or requesting certain (key,value)-pairs. It might have any number of reasons for doing this.

Misuse may be caused by source, forwarders or destination peers. The attacks perpetrated by a source peer would typically be structural or denial-of-service attacks. We have identified the problems of structural attacks in prior work [12], though specific solutions there remain open. Denial-of-service attacks are difficult, if not impossible, to distinguish from flash crowds (*i.e* [16]), and as such are generally handled through judicious load balancing. We do not address these problems further here.

Attacks perpetrated by destination peers can be of two types: dropping the request or replying with false data. While the second attack type is strictly harder to detect, in both cases the only correct solution is to replicate the data over disjoint destination peers. If replication is not used, then the only copy of the data is under the control of a malicious or faulty peer. Replication, however, is relatively effective if the replicas are randomly distributed among the peers, since such end-node attacks affect exactly as many peers as are malicious peers. Specifically, if p percent of the peers are malicious, then p percent of queries would either be unanswered or receive a false reply, presuming a uniform random distribution of queries. By replicating the data across multiple disjoint peers, and then sending each query to all such peers, the probability of being unable to distinguish a false reply from a correct one can be made arbitrarily low, being

$$\sum_{i=q}^{r} \binom{r}{i} p^{i} \left(1-p\right)^{r-i} \tag{1}$$

where r is the degree of replication ($r \geq 2$), and q is the quorum size ($2 \leq q \leq r$). That is, given r requests to r distinct peers, each holding the relevant data, q identical replies are required to accept that reply as authoritative. For small values of p, this can be approximated as $\binom{r}{q}p^{q}$. If we presume about 1% malicious peers, then a replication degree of three and quorum size of two would yield 0.03% of queries that would not have a quorum of correct replies. If malicious nodes collaborate, that 0.03% could include undetected false replies. If no collaboration occurs, then the query replies would simply fail to comprise a quorum, so no reply would be accepted.

Finally, attacks may be performed by forwarder peers. Such attacks are referred to as man-in-the-middle (MITM) attacks. Here the attack affects many more peers than there are malicious peers, as requests pass through multiple forwarders to the destination. In our prior work [12], we have shown that even a small number of malicious forwarding peers can harm the operation of the CAN. For example 1.5 % malicious peers in a CAN consisting of $10,000$ peers can corrupt 10 % of the messages transmitted. More precisely, the probability of an operation being attacked is $1-(1-p)^{l}$ where p is the percentage of malicious peers and l is the average path length to the destination. For low values of p (on the order of 1% or less) this can be approximated as lp, and thus we see that it is linear in the number of malicious peers. What is worse, however, is that the path length grows proportionally to $N^{\frac{1}{d}}$, where N is the number of nodes in the CAN and d is the dimensionality. Thus, the effect of malicious forwarders grows exponentially with the size of the CAN.

Clearly such MITM attacks pose a serious threat to the CAN, in particular, and to DHTs in general. The next question is, given that their ability to harm is far in excess of their number, what options does a forwarding peer have by means of attack, and how might those attacks be detected and/or mitigated. There are several possible MITM attacks a forwarding peer might employ:

1) It can simply drop the message.
2) It can modify the destination key.
3) The attached data could be modified.
4) It can claim to be the destination peer and answer the message with a spoof result.

The first three attacks are relatively easy to detect. For the first attack, the sender can regard the message as lost when a timer expires. This is likely necessary in a robust implementation of the CAN anyway, as a forwarding peer may leave at an inopportune moment. In order to detect attack 2 and 3 we propose to use handshaking. When an operation is received by the destination peer, it generates a hash sum of the received data. It then sends an acknowledgment message directly to the sender of the request containing the key and the hash sum. If any of these properties have been modified by forwarding peers it will be detected by the sender of the request in this step. Unfortunately the sender of the request message is unable identify the attacker, since each peer knows only its neighbors. Further, absent non-deterministic or alternate path routing, repeating the request is futile in attacks 1 and 2, and the message will again be routed through the malicious peer. Attack number 4 will go unnoticed if no countermeasures are employed. We refer to this attack as *masquerading*.

V. Countermeasures

Standard DHT single-path routing to un-replicated nodes has a serious MITM vulnerability. Given the various attacks we have just enumerated, it is clear that some mechanism is needed to address the problem. Further, we do not believe that the message-checking approach to the problem is sufficient, as it can at best identify that there is a problem, but provides no alternate route to the desired peer. Rather, we propose three replication-based forwarding protocols. All three approaches issue multiple copies of any given request over several, hopefully disjoint, paths to the destination(s). We then form a quorum of the returned results. This prompts the need for a general quorum method for the three replication approaches.

When r replica messages are issued, we may receive any number of replies. Specifically, we will have less that r replies if some MITM attacker drops a message, or more than r if it chooses to issue multiple replies to a single request. As such, the quorum protocol must distinguish replies, to ensure they are from unique request-message replicas, rather than duplicates. This is achieved through the use of nonces [14]. Each replica of the request is given its own nonce, which must be present in the reply. We term a a non-duplicated reply with a valid nonce as a valid reply. This effectively precludes a malicious peer from returning multiple replies to the same request, thus overwhelming the quorum protocol.

Second, given that malicious peers may drop requests, and given a desire to minimize the latency of requests, even in the presence of such replication, we do not wish to wait for all replies. Further, we presume that the presence of a dynamic protocol that can compensate for the existence of attackers will discourage attacks, and thus keep the number of malicious peers low. As such, we propose the following protocol. We initially use a small quorum size q (typically two or three). If the first q replies received are valid and identical, the reply is accepted. If the first q replies are not identical, we wait for the remaining replies, with a timeout based on the duration required to receive the first q replies. Having collected all replies, or timed out waiting for such, we then check to see if we have received $\lfloor r/2 \rfloor + 1$ valid and identical replies. If we have, it is accepted as the reply. If malicious nodes do not collaborate and replicated messages traverse disjoint paths, this approach should rapidly accept replies where there is no attack, while still effectively resolving those cases where attacks occur. When malicious nodes are expected to collaborate, the value of q may need to be raised, to ensure that q identical but false replies are not received.

Our three forwarding protocols each use this quorum algorithm to determine if a reply is valid. However, they differ in how they replicate the request message. The first protocol extends the concept of *multiple realities* [1] with simultaneous message replication. The *multi path routing* protocol is an enhanced version of our earlier work [12], and the *proximity routing* protocol is a new proposal that exploits the idea of a parallel random paths in order to generate a competent quorum. We now describe each of our protocols in more detail.

A. Multiple Realities

Given that we are addressing the MITM problem, rather than the end-node problem, it is sufficient to replicate queries alone, and two of our protocols follow that approach. However, our first approach instead extends the concept of multiple realities. As with the original concept, the key-space is replicated r times. Each peer joins in all r realities independently. As such, the zones of each peer in different realities are extremely likely to be at distinct and distant

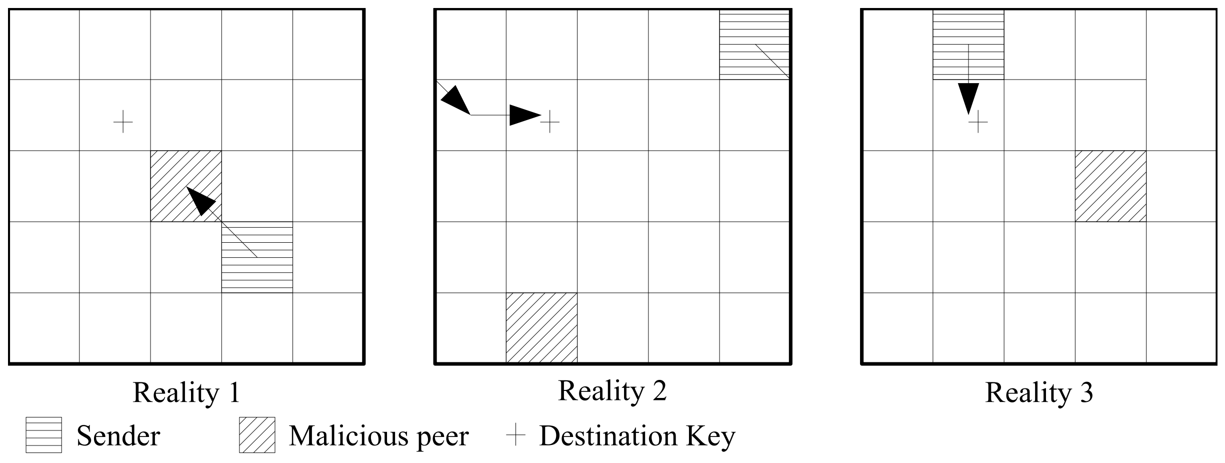


Fig. 2. Example: Routing in multiple-realities

locations in key-space coordinates, with each peer maintains a different set of neighbors for each reality. The peers now generate r messages for each request that are sent in parallel in all realities. Because of the different zone layout in each reality, the paths are likely to have different forwarders.[2]

A malicious peer can only attack a request in the reality where it is member of a path. Consider the example shown in Figure 2. It depicts an idealized, two-dimensional CAN, consisting of 25 peers that is replicated into three realities. Among these peers is a malicious peer that attacks incoming messages. The sender sends out a message for the key k in all realities. In the first reality the malicious peer is part of the path and attacks the message. In the two other realities the messages are forwarded without any attack. Thus, the sender can form a quorum and perform the request on these replicas of key k.

While this approach promises to be effective against both MITM attacks and end-node attacks, since both data and messages are replicated, we must observe that triplication will in general be insufficient. The reason is that MITM attacks have a much higher probability of occurring than do end-node attacks, being (as noted earlier) proportional to the path length and exponential in the number of peers. In this paper, we do not attempt to estimate the cost of maintaining a large data replication level as it is highly dependent on the update frequency, consistency requirements, *etc.* We anticipate, however, that it would not be desirable for systems with moderate or large update frequencies, particularly if consistency requirements are strong. On the more-positive side, this approach likely reduces the latency, particularly with a base quorum size of two. By

[2]Our simulation confirms that overlapping paths are rare incidents (*cf.* VII).

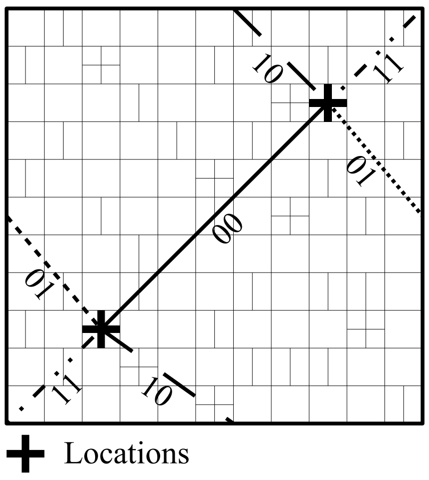


Fig. 3. Example: Possible Paths in a 2-Dimensional Torus Key Space

choosing paths in a shortest-path manner in each reality, some of the paths are likely to have a lower length than others. These paths can form the quorum and thus reduce the latency.

B. Multi-Path Routing

Our two remaining approaches avoid the expense of data replication by operating in a single reality, attempting to send replicated messages over distinct paths to a common destination. Our first technique for such message replication is multi-path routing [12]. It works based on taking advantage of the fact that the key space is organized as a torus. A message can reach each particular key from any point by either going the direct route or the by going in the opposite direction around the wrap[3] of the torus. This is possible in each dimension. Figure 3 shows exemplarily each possible routing path for a message from (0.3,0.3) to (0.7,0.7). We enforce or avoid a wrap of a dimension by attaching a bit-mask to the message that contains one bit for each dimension d in the key space. A peer that wants to issue a request for a particular key may issue up to 2^d replicas of the message. Each replica is marked with a different bit-mask, $[B_1, B_2, \ldots, B_d]$. If bit B_i is set, the message replica will cross the wrap in dimension i. Otherwise, the message replica traverses the torus without wrapping. Figure 3 shows how different bit-masks affect the choice of a path in a two-dimensional key space.

In contrast to the concept of multiple realities, this approach removes the overhead of data replication and avoids synchronization problems. In addition, if the data replication level is three,

[3]The shape of the torus results in a wrap from 0 to 1 and vice versa.

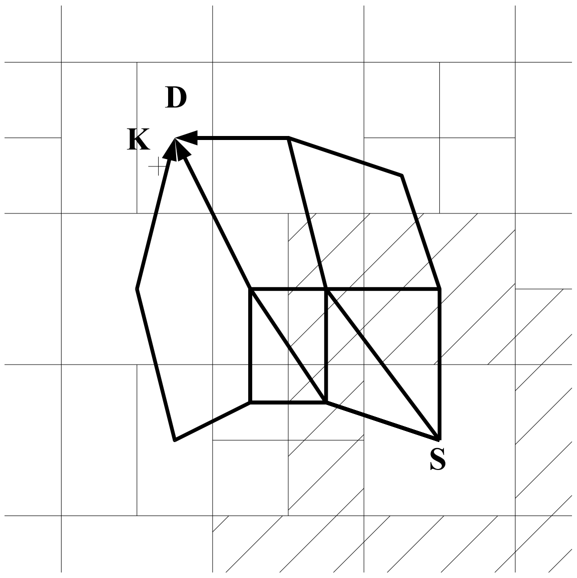


Fig. 4. Example: Fanout of Proximity Paths in a 2-Dimensional CAN

then multiple realities will be insufficient to deter MITM attacks, while a reasonably-dimensioned CAN (likely at least three or higher) will have significantly greater opportunities for message replication. However, compared to the classic routing algorithm, the paths are dictated by the bit-masks rather than the actual distance to the destination. As such, some messages travel always the longest valid path, and on average, path latency is doubled. This is particularly problematic, as the probability of a MITM attack is proportional to path length. On the more-positive side, by taking all pairs of opposite directions around the torus, the probability of paths being disjoint is high (though this becomes less true in higher-dimensional CANs).

C. Proximity Routing

Our third technique attempts to alleviate the problem of long paths to the destination that is implied in the multi-path routing method. We propose *proximity routing* to generate pseudo-random paths that can not be predicted by malicious peers, while still taking a short path to the destination. Here we loosen the greedy forwarding protocol of the classic routing algorithm, requiring merely that a message be forwarded to a *closer* neighbor, rather than the closest neighbor. The particular next hop is determined randomly from the set of all neighbors which are closer to the key. This modification enables the peers to send out multiple replicas of the same message along different paths. Figure 4 shows how the proximity paths fan out in a two-dimensional CAN.

Proximity routing comes with a protocol that is very easy to implement, and should avoid routing messages over long paths. The downside of the protocol is that paths cannot be assumed to be pairwise peer disjoint. Messages are forwarded along randomly selected paths, and thus we can give only stochastic guarantees that a particular malicious peer is not member of more than one route. This becomes a significant issue if a direct neighbor of either the sender or the destination peer is malicious. We suspect that when the caching-based routing of Buchmann and Böhm [17] is used, the paths would likely be disjoint. Finally, in contrast to multi-path routing and the multiple-realities approach we cannot guarantee that even one message travels the shortest route between sender and receiver.

VI. Formal Analysis

We provide an analytical way to describe the multi-path routing and the use of multiple realities in this section. The analytical models are useful to estimate the overhead and the impact of MITM on the CAN for different algorithms in order to choose the right configuration for the desired application. We start by introducing the average path lengths. Then we derive a model for the attack probabilities and overhead for each algorithm.

We use the following assumptions:

1) All peers have zones of uniform size.
2) The requests are sent to uniformly distributed keys and are all independent from each other. In addition, we assume that the peers which issue the requests are chosen with a uniform distribution.
3) The number of malicious peers is fixed. A malicious peer always commits attacks.
4) The size of the CAN is fixed.

In order to derive a simple probabilistic model, we assumed a regular zone layout. Through preliminary experiments, we discovered that the average path lengths for a CAN with a regular zone layout are slightly longer than in a realistic CAN. Thus, the length the estimation outperforms a real scenario. Therefore, the simulation provides an upper bound estimation. In order to ease our analysis we assume uniformly distributed destination keys and source peers. In the real-world, these distributions follow the requirements of the application on top of the DHT. However, we do not want to make assumptions on the application level. From out point of view, malicious behavior is an exception. We assume these attackers to attack any request.

If the actual behavior occurs sporadically or the actual number of malicious peers is below this bound then this assumption grants an lower bound reliability. In Section VII we will show that these assumptions do not affect the applicability of our model in reality.

The following parameters need to be given to build a model for each routing algorithm:

- The dimensionality d of the key space.
- The total number of peers n.
- The number of malicious peers m.
- The size of the quorum q.
- The number of messages u per request.

We begin by deriving the estimation formulas in Section VI-A, VI-B, and VI-C. Afterwards we provide an exemplary calculation in Section VI-D for a 4-dimensional CAN consisting of 83,521 peers that is comparable to our experimental evaluation in Section VII.

A. *Single-Path Routing*

In this section we derive a model for single path routing that can be used estimate the average path length, overhead, and attack probability.

The average path length can be approximated by $\frac{1}{4}$ of the distance between the two points that span the key space. For a d-dimensional CAN with regular sized zones consisting of n peers, the average path length $l_{sp}\left(d,n\right)$ can be calculated by Formula 2.

$$l_{sp}\left(d,n\right)=\frac{(dn)^{\frac{1}{d}}}{4}O_{sp}\left(d,n\right)=\frac{(dn)^{\frac{1}{d}}}{4} \tag{2}$$

The number of hops is equal to the generated overhead. Thus, this formula also estimates the generated overhead (*c.f.* Formula 2).

The attack probability can be estimated based on $l_{sp}\left(d,n\right)$. To corrupt a request at least one forwarder needs to be malicious. Given $l_{sp}\left(d,n\right)$ and m, the attack probability can be calculated by modeling each hop as a Bernoulli trial with a success rate, given by Formula 3. To corrupt a path at least one trial needs to succeed. Thus, the attack probability is given by Formula 4 and the reliability by Formula 5.

$$\begin{aligned} p_h\left(n,m\right) &= \frac{m}{n} & (3)\\ P_{sp}\left(n,d,m\right) &= 1-\left[1-p_h\left(n,m\right)\right]^{l_{sp}(d,n)} & (4)\\ R_{sp}\left(n,d,m\right) &= \left[1-p_{sp}\left(n,m\right)\right]^{l_{sp}(d,n)} & (5)\end{aligned}$$

B. Multiple Realities

Using multiple realities is a generalization of single path routing. The formula for the average path length for single path routing can also be used to calculate the average path length $l_{mr}\left(d,m\right)$ for this algorithm. Thus, $l_{mr}\left(d,n\right)) = l_{sp}\left(d,n\right)$. The probability for one path $p_{pmr}\left(n,d,m\right)$ to be attacked is equal to the overall attack probability for single path routing $p_{pmr}\left(n,d,m\right) = P_{sp}\left(n,d,m\right)$. We model each of u paths path as Bernoulli trial. A successful trial models an attacked path. To corrupt a request, at least $u-q+1$ trials need to succeed. Thus, the probability to corrupt a request can be calculated as follows.

$$\begin{aligned} P_{mr}\left(n,d,m,u,q\right) &= \sum_{i=u-q+1}^{u}\binom{u}{i}p_{pmr}\left(n,d,m\right)^{i}\left[1-p_{pmr}\left(n,d,m\right)\right]^{u-i} & (6)\\ R_{mr}\left(n,d,m,u,q\right) &= 1-P_{mr}\left(n,d,m,u,q\right) & (7)\end{aligned}$$

We estimate the reliability for this algorithm by Formula 7. The overhead is calculated by summing up the number of hops per path. The overhead can be calculated by Formula 8, using the average path length for single path routing.

$$O_{mr}\left(n,d,u\right) = u\cdot\ l_{mr}\left(d,n\right) \quad (8)$$

C. Multi-Path Routing

A peer that uses multi-path routing sends out multiple messages in one reality. This poses a challenge to our model description. Unlike using multiple realities, the path that are generated with multi-path routing are likely not to be peer disjoint. We approximate this feature by assuming that $n-1$ of n paths are independent. Thus, the choice of the quorum size needs to be at least 3 for being effective in detecting corruptions. However, a quorum of 2 might be sufficient to reduce the impact of man-in-the-middle attacks. But the reliability derived by our formal model will overestimate a realistic scenario. We derived the model for this routing algorithm analog to multiple realities by modeling each path as independent Bernoulli trial. The average path length for multi path routing differs from single path routing. We approximated the average path length by using half of the distance between the points that span the key-space, see Formula 9. Based on these Formulas and the hop attack probability (see Formula 3), the path attack probability can be calculated by Formula 10. Analog to the Formulas 6 and 7 the probability to corrupt a request and the system reliability can be calculated by the Formulas 11 and 12.

$$l_{mp}\left(d,n\right) = \frac{\left(dn\right)^{\frac{1}{d}}}{2} \tag{9}$$

$$p_{pmp}\left(n,d,m\right) = 1-\left[1-p_h\left(n,m\right)\right]^{l_{mp}\left(d,n\right)} \tag{10}$$

$$P_{mp}\left(n,d,m,u,q\right) = \sum_{i=u-q+1}^{u}\binom{u}{i}p_{pmp}\left(n,d,m\right)^i\left[1-p_{pmp}\left(n,d,m\right)\right]^{u-i} \tag{11}$$

$$R_{mp}\left(n,d,m,u,q\right) = 1-P_{mp}\left(d,n\right) \tag{12}$$

The overhead can be calculated analog to Formula 8 as follows.

$$O_{mp}\left(n,d,u\right) = u\cdot\ l_{mp}\left(d,n\right) \tag{13}$$

D. Example

After having introduced the formulas for each algorithm we will now calculate the attack probabilities for each algorithm using a configuration similar to our simulation configurations. We use a 4-dimensional CAN consisting of $83,521$ peers. The number malicious peers varies from 0 to 4000 with a step of 400. We have used 5 request messages for multiple realities and multi-path routing. We used a quorum size of 3. The results are shown in Figure 5.

Malicious behavior poses challenge to the CAN. Only 5 % malicious peers can corrupt about 25 % of all sent requests. Multi path routing reduces the impact of the attacks for a low percentage of malicious behavior at moderate cost. For this case it is not effective anymore if the malicious peer exceed 2.9 % of all peers. This is due to Formula 11 exeeds p_{sp} that is actually the base

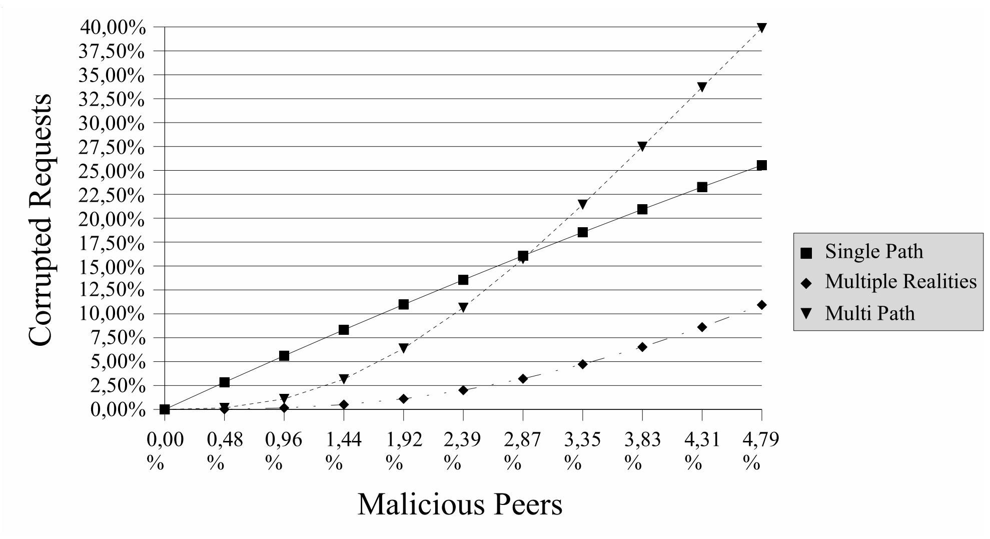


Fig. 5. Example Calculation for Single path Routing, Multi Path Routing, and Multiple Realities

for the multi-path calculation. These points can be calculated by solving $P_{mp}(n, d, m, u, q) = P_{sp}(n, d, m)$ for selected (u, q)-combinations. In contrast to multiple realities multi path routing does not need to replicate the key-space. However, if consistency is a hard requirement for the application the use of multiple realities is necessary.

VII. Simulation

After having analyzed the basic properties by calculation, we analyze malicious behavior for different ratios of malicious peers by simulation. We simulated the concepts of multiple realities, multi-path routing, and proximity routing in order to answer the following questions:

- How do the proposed protocols compare to the unmodified proposal of the CAN?
- What are the differences with respect to the attack resistance of the protocols?
- Which protocol comes with the least overhead in comparison to the original CAN protocol? To be more precise, what is the amount of network traffic caused by the various protocols?
- How do the protocols affect the *latency* between issuing a query and obtaining the query result?

In order to answer these questions, we ran a set of simulations on a cluster of Linux Workstations. We used *cansimulator* (available through [18]), a CAN-implementation of our own that can be extended with the routing protocols proposed. The setup of each of our experiments contained 83,521 peers organized in a static four-dimensional key space. During the simulation no peers left or joined the CAN, and all peers have equally sized zones. We simulated multiple

Algorithm	number of paths or realities	size of quorum	obtained reliability	generated overhead
Single path routing	(1)	(1)	93.35 %	6.76
Multiple realities	5	2	99.99 %	33.81
Multi path routing	5	2	99.27 %	60.44
Proximity routing	5	2	98.01 %	81.46

Fig. 6. Configurations for 95 % Reliability for 1.2 % Malicious Peers

realities by simply replicating the key space r-times. The peer identifiers in each replica have been permuted to ensure that the peers manage different zones in different realities.

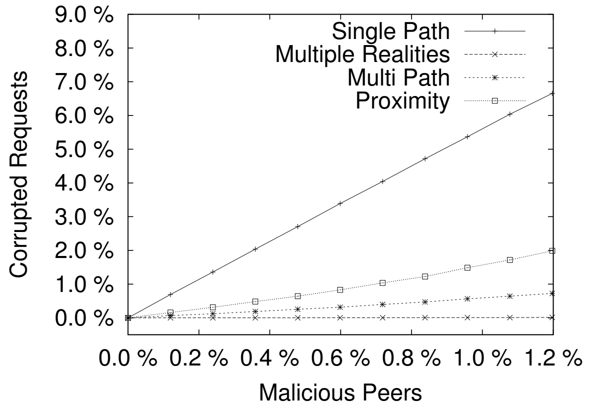
For each experiment we simulated 1,000,000 requests. The peers that issue the requests were randomly chosen, and the destination keys of the requests were equally distributed over the key space. In this setup we simulated up to 1000 malicious peers which were equally distributed over the key space as well. In order to avoid side effects from a stochastic formation of peers, we generated a new distribution of malicious peers for each sequence of request messages.

In order to obtain meaningful simulation results, we have to determine useful external parameters for our simulation. Such external parameters are the size of the quorum, the numbers of parallel realities or paths, the number of malicious peers and the number of corrupted requests that are acceptable to the participants. We think that a number of up to 1,000 malicious peers (1.2% of the absolute number of peers) which try to attack the CAN in parallel is a conservative assumption.

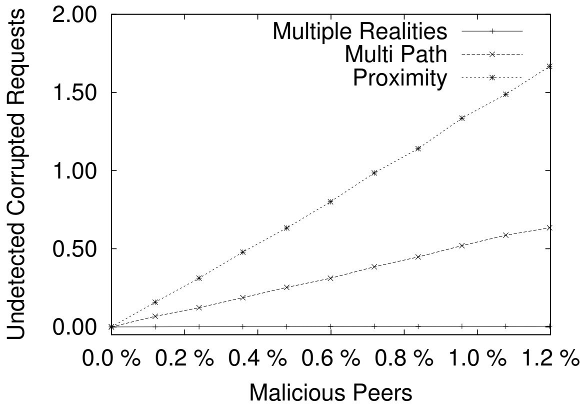
DHTs are not intended to drive applications that depend on transactional guarantees, hence we assume a number of at most 5% corrupted requests can be accepted (*i.e.*, we demand a reliability of at least 95%). Each of our proposed countermeasures satisfy this requirement at a quorum size of 2 and a number of 5 parallel realities or paths. If not otherwise stated, we will use this setting for our experiments. Table 6 shows the overhead and the obtained reliability for our countermeasures for the chosen configurations. We obtained these configurations by preliminary simulations. We increased the number of paths until a level of 95% reliability was reached. For comparison, we also included the measured values for single-path routing.

A. *Impact of the Attacks*

At first we will show how our countermeasures adapt to an increasing number of malicious peers. Therefore we use the simulation setup described so far, and vary the number of malicious peers from 0 to 1000 with an increment of 100. Figures 7(a) and 7 show the results of a series of experiments for each of the protocols.



(a) Total Corrupted Requests



(b) Undetected Corrupted Requests

Fig. 7. Corrupted Requests

Figure 7(a) shows how many requests are corrupted while being forwarded.

Using multiple realities adapts best to an increasing number of malicious peers. 1.2 % malicious peers only corrupt 0.013 % of the requests sent. This is a clear advance compared to single path routing. Multi path routing performs at 0.726 % corrupted requests that is an 89 % reduction of the impact of MITM. Proximity routing does not perform as well as the other countermeasures do, but also reduces the impact of MITM by 70 %.

Diagram 7(a) shows recognized and unrecognized attacks. Obtaining a quorum does not necessarily mean that a request was processed correctly. Two or more independent request messages may be forwarded to the same malicious node that can attack the message. Figure 7(b) shows how many corrupted requests would have gone undetected in the worst caste of our setting, normalized to all request sent. Multi path routing performs worst in this category. The peers presume approximately 87 % of all corrupted requests to be correct at a rate of 1.2 % malicious peers. Proximity routing performs at a rate of 84 % at the same rate of malicious peers. The peers demand a quorum of 2 independent answers for each request. Obviously, by using single path routing no corrupted request will be detected on the protocol level. In the case of the other protocols, there can be overlapping paths where one attacker would be able to attack two or more messages. To model the worst case, we assume that a malicious peer that attacks multiple messages of the same request act as masquerading peer. The probability of undetected attacks is different for each of our protocols. In general, we assume that replies sent by malicious nodes that masquerade the destination peer arrive earlier than replies sent by the correct peer, since the attackers are closer to the sender. This assumption also increases the probability of overlooking corrupted requests sent by masquerading peers.

Another reason for this problem is the quorum size of two for proximity and multi path routing. We expect that the paths are likely to overlap in proximity routing. A peer close to the source or destination peer can corrupt most of the request messages sent. For multi path routing this is only an issue for a small quorum size, since the paths span across all cardinal points of the key space, as seen in Figure 3. Increasing the size of the quorum reduces this problem. However, the straightforward solution to simply increase the quorum size does not solve the problem: The number of corrupted requests will also increase, since a sender might not be able to form a quorum from the responses anymore. This correlation is shown in Figure 8 for multi path routing, and shown in Figure 9 for proximity routing.

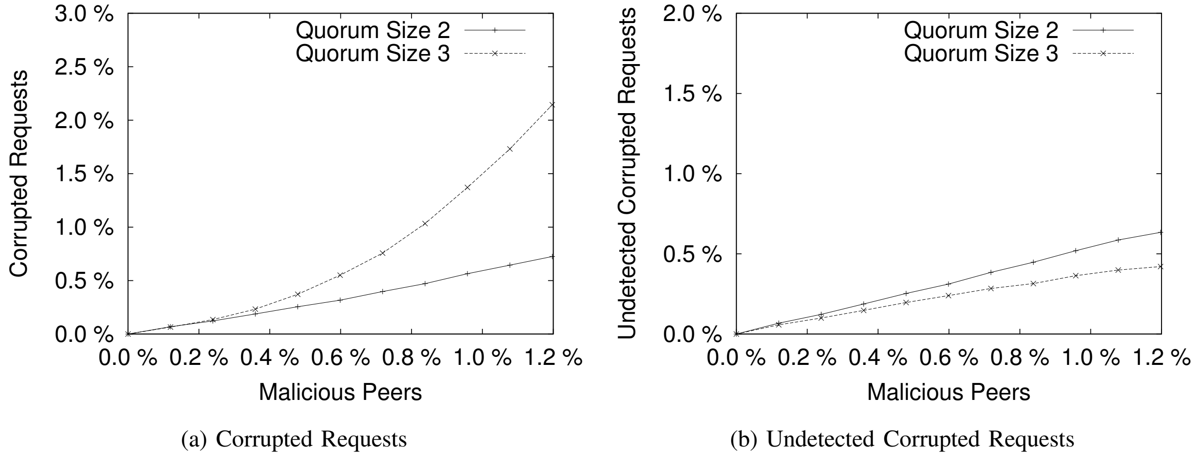
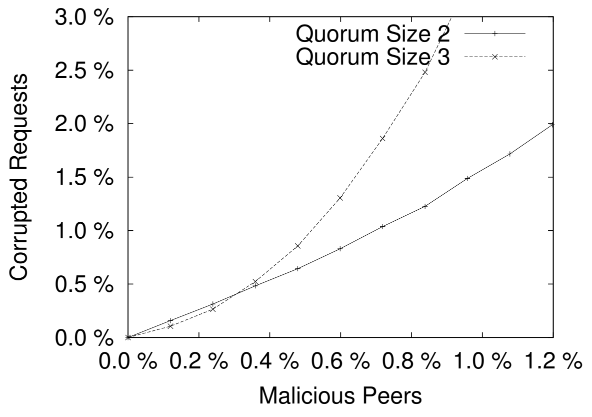


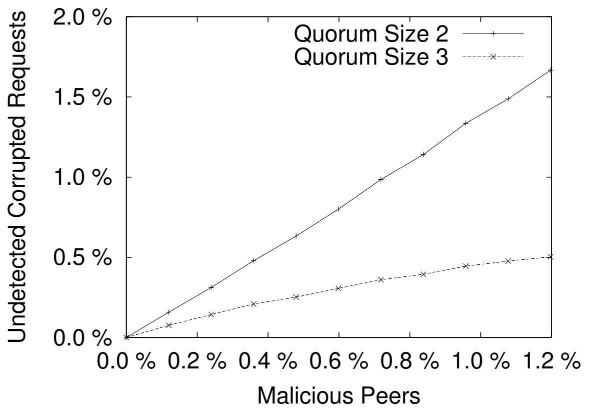
Fig. 8. Multi Path Routing: Corrupted Requests vs. Quorum Size

The overall reliability for processing a request correctly is minimal for a quorum size of 2. However most of the corruptions are not detected. For a quorum size of 3 the senders of the request messages detect the majority of the attacks while the overall rate of corrupted requests remains reasonably small. However, proximity routing does not satisfy our 95 %-reliability requirement for 1.2 % malicious nodes. 5.1 % of the requests sent are corrupted using a quorum size of 3.

Increasing the quorum size to 3 even improves the overall routing reliability for a small number



(a) Corrupted Requests



(b) Undetected Corrupted Requests

Fig. 9. Proximity Routing: Corrupted Requests vs. Quorum Size

of malicious peers using proximity routing. The ratio of missed corrupted requests also scales similarly to multi path routing for an increased quorum size.

B. Overhead

In this section we analyze the overhead that is generated from each of our proposed countermeasures. Analyzing the number of hops made for each request is necessary in order to compare the bandwidth efficiency of the algorithms. In this section we do not study either the storage or message overhead that is required to maintain multiple realities. While it is clear that the storage costs are proportional to the number of realities used, the message overhead required to maintain synchronized replicas is a function of the consistency required, the protocol used, the degree of updates, *etc.*, and is thus beyond the scope of this study.

Again, we use the introduced setup for our simulations. Since we regard malicious behavior as an exception, we compare the performance of the algorithms without malicious peers. Figure 10 shows the overhead for our proposed configurations. It indicates that the overhead is increasing by using any of our countermeasures. The overhead generated by using multiple realities is relatively low compared to the other countermeasures, but is also four times higher than single-path routing. This is the result of using four realities. Multi-path routing generates ten times as much overhead as single-path routing. This is the result of the varying paths lengths of the request message. On average the path for each request message is twice as long as the path that would have been generated by single path routing. Proximity routing generates a further 30 %

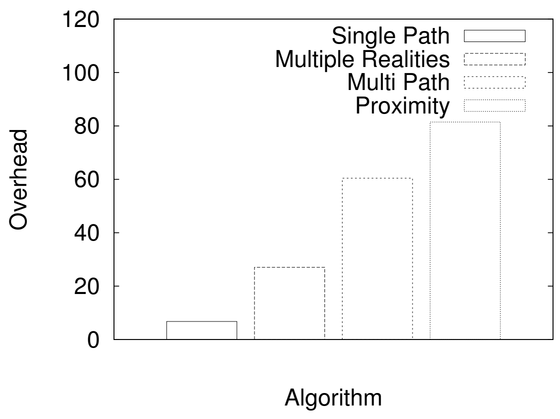


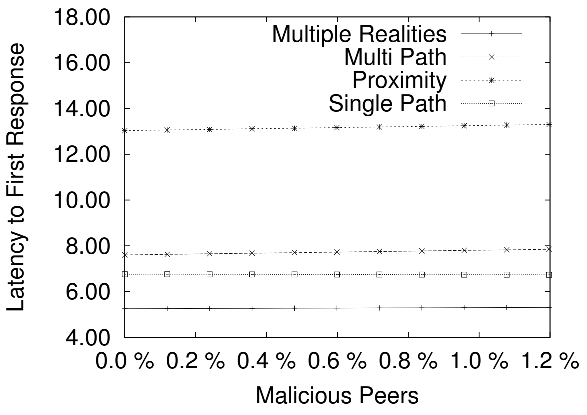
Fig. 10. Overhead Generated for Each Algorithm

more overhead than multi-path routing. Using proximity routing results in longer path lengths than multi-path routing on average. As noted earlier, we suspect that in CANs that use caching in their routing protocol, proximity-based routing will produce more promising results.

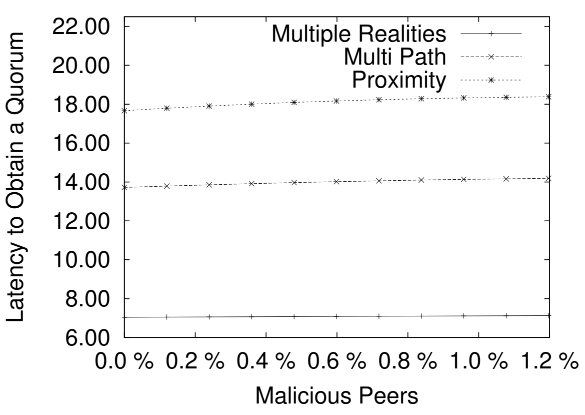
C. Latencies

The latency to perform a request is an important criteria for a CAN routing algorithm. When a peer sends a message replicated over multiple paths or realities, the latency depends on the size of the quorum q. We assume that the messages can be sent in parallel, thus the *latency to obtain a quorum* (LTQ) is equal to the latency of the first q-quantile of all paths for a request. Since the latency for unanswered requests is infinite, we only measure the LTQ for requests where the number of answers is sufficient to form a quorum. We cannot obtain a quorum in the case of single path routing or unanswered messages. Thus, we introduce the *latency to the first response to arrive* (LTF) as the delay between issuing a request and obtaining the first answer.

Figures 11(a) and 11(b) graph LTF and LTQ for our protocols. Figure 11(a) shows an average LTF of 6.7 hops for single path routing that is independent from the percentage of malicious peers. Using multiple realities results in a smaller LTF than single path routing. Since the senders zones are at different locations in different realities, one of its zones is likely to be closer to the destination key (cf. [1]). Multi path routing has a slightly increased LTF compared to single path routing. The evaluation for proximity routing shows that the LTF is almost twice as high as it is for single path routing. The latency slightly increases with a growing number of malicious peers for our proposed countermeasures. Since multiple messages are sent, the presumably shortest path is more likely to be attacked when the number of malicious peers



(a) Latency to the First Response (LTF)



(b) Latency to Obtain a Quorum (LTQ)

Fig. 11. Latency Evaluation for Different Routing Algorithms

increases. After having compared the ability of the proposed algorithms to deal with faulty peers, we compare their performance in countering man-in-middle attacks that involve modifications of messages, masquerading, as well as dropping the messages. In this case the sender needs to form a quorum of a size q from the responses in order to carry out an operation. Figure 11(b) shows the latency to quorum evaluation the proposed countermeasures. Because the sender has to wait for q messages to arrive before performing the request, the latency increases significantly for the proposed countermeasures. The increase of the latency for multi-path routing is higher compared to proximity routing. The paths of the request messages for one request have very varying lengths for multi-path routing. Thus, attacking a short path results in an high increase of the latency.

VIII. Discussion

Based on our analysis and simulation results, we are now in a position to discuss the situations where our proposed countermeasures are applicable. For any application in which data corruption is a problem, replication will be essential, and thus some level of multiple realities will be required. However, the greater the degree of replication employed, the greater the cost in ensuring consistency across those replicas. While, it is undeniable that the multiple-reality approach provided the lowest latency and bandwidth overhead and the highest reliability, we believe that any cases in which update is moderate or frequent will incur costs in maintaining consistency that outweigh the advantages of replication if the degree of replication is large enough to mitigate

MITM attacks, rather than simply precluding end-node attacks.

Multi-path routing, while yielding a lower reliability than the use of multiple realities, still reduces the impact of MITM attacks substantially compared to single-path routing. Further, it does not incur significant cost or complexity for update operations, as does the use of multiple realities. Finally, it can readily be dynamically adjusted in replication degree, up to 2^d, which will typically be a significantly higher level of message replication than will be achievable in practical multiple-reality settings. As such it can dynamically adapt to changing circumstances, based on the perceived threat level.

Proximity routing does not improve the reliability of the system substantially for the presented cases. It has a relatively high overhead compared to the other algorithms. However, proximity routing forwards requests along randomized paths. Therefore it would be more difficult to guess a path and to attack a certain message. On the other hand, it would be difficult to evade peers adjacent to the destination, which are likely to be selected, even with cache-based routing.

Given the trade-offs between multiple realities and single-reality message replication, we propose a blended approach, in which a low degree of data replication is used, as is needed to preclude end-node attack, and this is then augmented with multi-path routing as required based on the measured threat level. The precise approach to take in this case is the subject of future work.

IX. Conclusion

In this paper we have demonstrated that man-in-the-middle attacks are a significant threat that have to be faced by every DHT. The nature of DHTs (*i.e.*, the lack of global knowledge and central control) makes this a difficult problem to tackle.

In this paper we have quantified the threat of both end-node and MITM attacks, and shown that MITM attacks are represent a major vulnerability in DHTs in general and CANs specifically. We present three replication-based countermeasures that promised to reduce the impact of MITM substantially by exploiting the concepts of random path selection, multiple realities, and message replication, comparing the approaches both by formal analysis and by simulation. We have shown that it is possible to significantly reduce the impact of MITM attacks at reasonable cost.

Many issues remain open for future work. First, we wish to develop a blended protocol, combining multiple realities with multi-path routing. As noted, we believe this will provide the best trade-off in terms of update cost, dynamic response to threat, and MITM-attack mitigation. To do so we will need to incorporate update protocols into out system. Second, we plan to extend our analysis to other attacks, some of which are mentioned in this paper, such as denial-of-service attacks, and others, which we have enumerated elsewhere [12]. Finally, we plan to adapt our experiments to other DHT variants, with the hope of developing a generalized analysis that enables us to predict the performance of our countermeasures with any DHT.

REFERENCES

[1] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A scalable content-addressable network." in *SIGCOMM*, 2001, pp. 161–172.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[3] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. Springer-Verlag, 2001, pp. 329–350.

[4] R. C. Emil Sit, Josh Cates, "A dht-based backup system," MIT Laboratory for Computer Science, Tech. Rep., August 2003. [Online]. Available: http://project-iris.net/isw-2003/papers/sit.pdf

[5] R. Huebsch, J. M. Hellerstein, N. L. Boon, T. Loo, S. Shenker, and I. Stoica, "Querying the internet with pier," in *Proceedings of 19th International Conference on Very Large Databases (VLDB)*, September 2003. [Online]. Available: citeseer.ist.psu.edu/huebsch03querying.html

[6] D. Tam, R. Azimi, and H.-A. Jacobsen, "Building content-based publish/subscribe systems with distributed hash tables." in *DBISP2P*, 2003, pp. 138–152.

[7] E. Buchmann and K. Böhm, "Fairnet - How to counter free riding in peer-to-peer data structures," in *Proc. of the International Conference on Cooperative Information Systems 2004, Agia Napa, Cyprus*, Oct. 2004.

[8] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management*. ACM Press, 2001, pp. 310–317.

[9] A. Wagner, T. D&#252;bendorfer, B. Plattner, and R. Hiestand, "Experiences with worm propagation simulations," in *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*. New York, NY, USA: ACM Press, 2003, pp. 34–41.

[10] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002, pp. 251–260.

[11] A. Singh, M. Castro, A. Rowstron, and P. Druschel, "Defending against eclipse attacks on overlay networks," in *Proceedings of the 11th ACM SIGOPS European Workshop*, Leuven, Belgium, September 2004.

[12] T. Reidemeister, K. Böhm, P. Ward, and E. Buchmann, "Malicious behavior in content-addressable peer-to-peer networks," in *Proceedings of the third Annual Communication Networks and Services Research Conference*. IEEE Computer Society, 2005, pp. 319–326.

[13] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002, pp. 261–269.

[14] W. Stallings, *Cryptography and Network Security: Principles and Practice, 2nd Edition.* Prentice-Hall, 1998.

[15] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks," 2002. [Online]. Available: citeseer.ist.psu.edu/castro02secure.html

[16] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites," pp. pages 252–262, May 2002. [Online]. Available: citeseer.ist.psu.edu/jung02flash.html

[17] E. Buchmann and K. Böhm, "Effizientes Routing in verteilten skalierbaren Datenstrukturen," in *Proceedings der 10. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web, 26. bis 28. Februar 2003, Leipzig, Germany*, 2003.

[18] T. Reidemeister, "Cansimulator," 2005. [Online]. Available: http://cansimulator.sourceforge.net

Thomas Reidemeister <treideme@student.uni-magdeburg.de>

Thomas Reidemeister studies computer science in engineering at the University of Magdeburg, Germany. He is expected to receive his diploma in February 2007. His research interests are the reliability of peer-to-peer structures and communication systems that arose from a visiting scholarship at the University of Waterloo in 2004/2005. He works as student assistant for Klemens Böhm since April 2003.

Klemens Böm <boehm@ipd.uka.de>

Klemens Böhm is professor (chair of databases and information systems) in the computer-science department at Universität Karlsruhe (TH). He has received his Diploma in Informatik and his Ph.D. degree (Informatik) from Technical University of Darmstadt. Before joining Karlsruhe University in 2004, he has been a professor at Magdeburg University. Prior to that, he has been affiliated with ETH Zurich and GMD Darmstadt. His research interests are distributed information systems, e.g., Peer-to-Peer systems and Grid infrastructures, data management in ubiquitous environments, and data warehousing. Klemens puts much effort in interdisciplinary research and application-oriented projects, currently ranging from biosystematics to traffic-data management.

Erik Buchmann <buchmann@iti.cs.uni-magdeburg.de>

Erik Buchmann received the diploma in business information technology from the University of Magdeburg, Germany. Currently he works as a PhD student and research associate at the Institute of Technical and Business Information Systems at the University of Magdeburg. His main research interests are Peer-to-Peer Data Structures and their optimization.

Paul Ward <pasward@ccng.uwaterloo.ca>

Dr. Paul A.S. Ward is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Waterloo, where he teaches senior undergraduate and graduate courses in networks and distributed systems. His research interests span the areas of distributed systems and computer networks. In distributed computing his work focuses on distributed-application management, and more generally on dependable and self-managing distributed systems. In networks his interest lies in wireless data networks, and more particularly in ad hoc, wireless mesh, and delay-tolerant networks. Prior to pursuing his Ph.D. he worked in both the hardware and software industries, covering the range from electronic-parking-meter design to developing the fast parallel load utility for the DB2 database system. Dr. Ward has a BScEng in Electrical Engineering from the University of New Brunswick and a Ph.D. in Computer Science from the University of Waterloo. He is a member of the IEEE, including the Computer and Communications Societies, as well as a Professional Engineer.