Long-Range CAN: to Enhance the Performance of Content-Addressable Networks

Balázs Kovács, Rolland Vida
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
Magyar tudosok krt. 2, Budapest, Hungary 1117
{kovacsb, vida}@tmit.bme.hu

Abstract

Distributed Hash Table (DHT) algorithms structure peer-to-peer networks to provide nodes with fast and scalable lookups. In recent DHT solutions, such as Chord and Kademlia, the contacts of a node in the overlay network are determined so as to keep up with a lookup cost of $O(logN)$ in a network of N nodes. As opposed to these, one of the first DHT solutions, called Content Addressable Network (CAN), has the drawback of limiting the lookup cost only in $O(dN^{\frac{1}{d}})$ where d is the number of dimensions in the coordinate space, a fixed network parameter. However, CAN has several merits to exploit, such as its multi-dimensional ID space and its special ID space structure. Thus, in this paper we present an improved algorithm called Long-Range CAN (LR-CAN), able to eliminate the rigidity of the original system and to provide a more scalable and resilient solution, not only compared to the original version, but also to the currently best performing DHTs that we already mentioned.

Keywords: Distributed Hash Table, Content-Addressable Network, small world, lookup-cost limitation, signaling optimization

1. Introduction

Distributed hash table algorithms construct structured P2P networks to control the communication cost of resource lookups. Resource or content lookups in DHT protocols are performed based on a key that is a hash print of the searched content. Nodes and keys are mapped on an identifier (ID) space, and the distance between the key and the node performing the lookup is determined. Usually, the protocols employ a greedy forwarding mechanism to forward the lookup to the owner of the key.

During this greedy forwarding, a node thus scans its contacts to find the closest node to the key. Generally, contacts can be classified into two categories: short-range and long-range contacts. The former category consists of contacts very close to the node in the ID space; they are indispensable for a node to properly participate in a DHT. If some, or all of the short-range contacts fail, a node may be unable to forward lookups. As opposed to these, long-range contacts are not mandatory for nodes to survive; however, they are useful to accelerate the lookups so as to be of a length proportional to the logarithm of the network size. DHT protocols define, set up, and maintain their short- and long-range contacts differently.

One of the first published DHTs was CAN [1]. It maps nodes and keys onto a d-dimensional ID space that wraps to a d-torus. The ID space is split into d-dimensional zones which are assigned to the nodes. A node is responsible for keys which map into its zone. The algorithm defines only short-range contacts for a node, which are its immediate neighbors on the torus. This concept significantly reduces the cost of maintaining contacts and thus signaling, as the number of contacts remains $O(d)$. However, lookup lengths may grow long and present large variations. An answer to this problem might be to increase the number of dimensions of the torus. As a consequence, the number of short-range contacts increases, the ID space is distributed along more dimensions, and thus hop distances decrease. The number of dimensions d can be chosen so as to make lookup lengths proportional to $logN$; however, d is a fixed system-wide parameter, and if one would like to change it on the fly, costly algorithms would be needed to re-hash and reshape the network accordingly. As a result, lookup lengths remain $O(dN^{1/d})$ [1]. If N increases significantly during the lifetime of the system, and if d cannot be modified accordingly, lookup lengths will be notably longer than with an $O(logN)$ system. Another solution to reduce lookup cost is the use of "realities". In each reality, the ID space is distributed randomly; thus, realities assign different zones for nodes and

introduce redundancy in the system, as nodes will probably store different keys in different realities. This method also reduces lookup cost, as lookups can jump between realities. However, realities are less effective in decreasing lookup lengths than the use of an increased number of dimensions.

Chord [2] maps nodes onto a ring as ID space. Short-range contacts of Chord are called successors and predecessors. Besides these, Chord also uses long-range contacts, called fingers, to keep lookup lengths $O(logN)$. In case of m-bit long identifiers, maximum m fingers are maintained, placed at exponentially increasing distances from the node. If we have a look at the basics of the system, Chord and CAN are very similar. A one dimensional CAN is an identifier circle, just as Chord. The main difference is that Chord has a different join and zone assignment strategy than CAN, and it has fingers, while CAN may have multiple dimensions. Unfortunately, the main drawback of Chord is the one-dimensional ID space and the unidirectional circle. Bidirectional Chord [3] eliminates the problem of unidirectionality, but increases the maintenance load of the network to keep fingers precise, although the bidirectional ring makes the system less sensitive to the imprecision of fingers.

Kademlia [4], a system developed from the basics of Pastry [5], is different in many ways. First, it defines a new distance metric, called XOR distance. The ID distance of two nodes is the XOR value of their identifiers. The XOR distance is unidirectional. In order to find short- and long-range contacts, each node examines the senders of messages which flow in the system upon joins and lookups, and decides whether to record a sender as a contact or not. In case of m-bit IDs, a Kademlia node n stores m buckets. Kademlia only defines ID ranges per buckets which should be covered by some nodes in contrast to the deterministic approach of Chord. To enhance routing performance, at most k nodes can be recorded to each bucket. A node stores a key if it cannot forward it to any node closer to the key. The parameter k also introduces redundancy by storing keys at multiple nodes. Kademlia has an iterative lookup strategy that always returns lookups to the initiator after visiting a new hop, until the destination is found. Compared to a recursive strategy it costs more messages, but may provide auxiliary information that may improve the lookup protocol, which is necessary in Kademlia as it learns the topology from lookup traffic.

Overall, most DHTs achieve route lengths proportional to $O(logN)$ if, in some way, they use long-range connections. In 1999 Jon Kleinberg worked out a network model for the small-world phenomenon which sets up requirements for decentralized systems to be able to find the "short paths" in the network. Based on the model of Watts and Strogatz [6], which proposes to have many short-range and a few long-range connections, Kleinberg determined a stochastic model of choosing long-range contacts in a network of arbitrary dimensions [7]. He stated that should the nodes of a 2-dimensional network follow his distribution of placing long-range connections (only one per node), the number of hops will be at most $O((logN)^2)$. Since most of the DHTs suit the Kleinberg model and have stricter rules of placing long-range connections, they obviously provide better path lengths in average than the one mentioned by Kleinberg.

Among the well-known DHT solutions CAN is the most similar to Kleinberg's general model as CAN defines operations in multiple dimensions. Its only shortcoming is not using long-range connections; thus, the variance of lookups yields a high value. An earlier paper presented an algorithm which enhanced CAN to be compliant with Kleinberg's methods, as it installed one "shortcut" per node according to Kleinberg's formula [8]. The authors achieved significant improvements in lookup lengths when compared to the original CAN; however, the approach is still not competitive with the performance of Chord or Kademlia.

The significance of the results presented in this paper is twofold. First, we present Long-Range CAN (LR-CAN), an algorithm that enhances CAN by utilizing long-range contacts, a solution very popular for scalable DHTs. Second, the algorithm introduces adaptivity to network size through a cost-limit function (denoted as $SR(N)$ further on). In contemporary DHTs the number of long-range contacts implicitly changes as network size changes, and their number is proportional to the logarithm of network size. On the contrary, LR-CAN controls the number of long-range contacts explicitly through the above mentioned definite cost-limit function. With the appropriate cost-limit function we can set LR-CAN so as to outperform Chord and Kademlia. Nevertheless, its multi-dimensional ID space, bidirectional along each dimension, and its special node mapping algorithm allow LR-CAN to keep the necessary maintenance traffic lower than in the case of the above mentioned state-of-the-art DHT protocols.

The rest of the paper is organized as follows. Section 3 presents the adaptive algorithm we propose (LR-CAN), while section 4 presents our theoretical expectations concerning its efficiency. Section 5 describes our simulation technique and presents the results by comparing them to the theoretical expectations and to the efficiency of alternative DHT solutions. Then, section 6 presents a method to minimize the signaling of LR-CAN. Finally, in section 7 we conclude the paper.

2. CAN Overview

As mentioned already in the introduction, in CAN each node is responsible for the resources which map into its zone. When joining the system, the new node draws a point

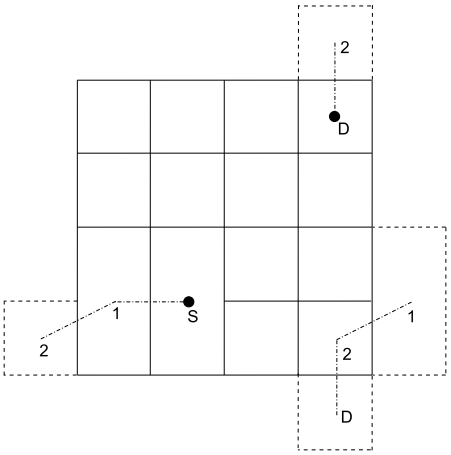

Figure 1. Routing on a 2-torus

P in the d-dimensional coordinate space according to a uniform distribution. The lookup for P is initiated by a gateway node. The node responsible for P will be the host for the newcomer; it will split its zone in two, based on the space splitting rules, it will notify the newcomer about its neighbors, and its neighbors about the newcomer. According to the space splitting rules, always the longer edge needs to be halved, but if more edges are of equal lengths, the edge with the lower-order dimension has to be split. Two nodes are neighbors if the coordinate spans of their zones overlap along $d-1$ dimensions and abut along one dimension. If the ID space is distributed uniformly to zones exactly with the same size, then each node will have $2d$ neighbors.

When a lookup is initiated, key K is deterministically mapped to a d-dimensional point P. Forwarding can happen only between neighbors. Each node checks its Euclidean distance to P and forwards the lookup to its neighbor closest to P in a greedy way. Note that CAN is bidirectional along each dimensions and the ID space wraps. For instance, the distance on a 1-torus between point 0.1 and 0.9 is 0.2 rather than 0.8. Fig. 1 shows a routing scenario in two dimensions on a 2-torus, where a lookup goes from S to D.

In case of a graceful leave, i.e., when a node is able to hand over his tasks before leaving, the leaving node has to choose one of its immediate zone neighbors to merge with its zone. If merging zones is unfeasible in the current state, since the zones would compose an invalid concave zone that contradicts the space splitting rules, the neighbor with the smallest zone will take over; thus, this node will temporarily own two zones, until assigning the zone to a new node or merging it with a third zone. Node failures are handled differently, as in this case the peers sensing the absence of another peer have to arrive to a consensus about the node that takes over the zone of the failed peer. We do not present the way of failure handling because it does not affect our algorithm. For more details about the operation of the original CAN algorithms, please refer to [1] and [9].

3. The LR-CAN Algorithm

LR-CAN uses long-range contacts to reduce lookup cost. The main difference between LR-CAN and other DHTs that employ long-range contacts is that the number of long-range contacts of a node changes adaptively according to a desired lookup cost, expressed by a cost-limit function, as defined in Section 4. In other DHTs the number of long-range contacts is an implicit and not controllable function of network size; thus, the algorithm and its current parameter set up implicitly determines the achievable lookup performance of the DHT.

The main idea behind the LR-CAN algorithm is that a node is able to assess the network size (number of nodes) individually, without the help of an information server. The reason for that is the distribution of the ID space that converges to a distribution with equal zones; this is because on a stochastic basis always the largest zones are split when nodes join the network and draw a random d-dimensional coordinate to determine their host node. In fact, the algorithmic principle behind this feature is that the node ID and the zone assigned to the node in the CAN ID space are independent. In Fig. 2, we can see the difference between the ID space assignment strategy of CAN and Chord. We insert three nodes in the same sequence for both systems. Let us assume that the hash of the IP address in Chord equals the random P coordinate the same node draws in CAN. On the left hand side we can see how a one-dimensional CAN assigns portions of the ID space to these three node. In CAN, the sequence of nodes joining the system affects the mapping. In this example, first, node A owned the whole ID space before node B joined. When B joined, the ID space was split into two equal parts, and the zone closer to the origin was preserved for node A. The same happens when node C draws the point mapping into the zone of node A. In contrast to CAN, in Chord, the sequence of nodes does not count, only their ID; they get the portion of the ID space where they are successors. As a result, the sizes of ID space portions in Chord depend on the hash implementation and the nodes joining the system, whereas in CAN space portions are deterministically balanced.

The aim is to limit the problematic original routing scheme of CAN with a cost-limit function, denoted as $SR(N)$. This function upper bounds the average lookup cost and triggers LR-CAN to increase the number of long-range contacts in the network. Obviously, the cost-limit is a function of the network size N. Since defining their own long-range contacts is the individual responsibility of the overlay nodes, and N varies, the nodes have to be able to

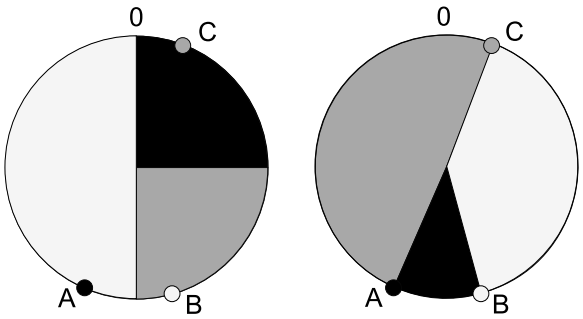

Figure 2. CAN and Chord ID space assignment strategy

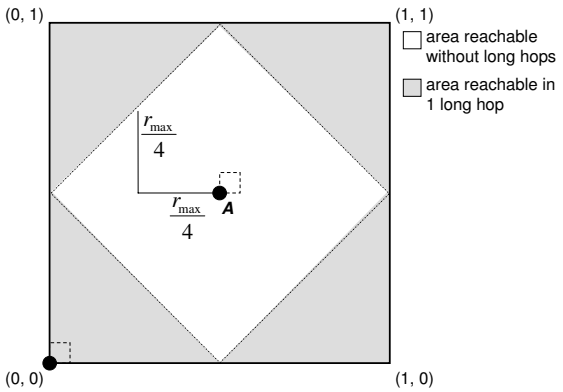

Figure 3. One long-range level is active

assess the network size N. Based on the assumption that the ID space distribution converges to equal zones, network size ($\widehat{N}$) can be assessed in several ways. One technique could be to calculate the sum of the zone sizes of immediate neighbors, and thus infer to network population. Low cost is the primal advantage of this solution as a CAN node always stores information about the zones of its neighbors. The drawback is that the result will reflect a local view of the CAN network; even if we assume a convergence to nearly equal zones, there will be local deviations that may provide inaccurate information about network size. A more accurate solution is by routing to the most distant point in the ID space and measuring the hop distance ($\widehat{R}_{max}$). We can estimate N through a formula that describes the connection between $\widehat{R}_{max}$ and N; just like the first technique, this solution also assumes a CAN ID space with equal zones. As this approach traverses a more significant portion of the ID space, it can provide more accurate information about N than sampling the zone sizes of immediate neighbors. Unfortunately, this second technique has higher cost, because of the necessity to route to the most "distant" point; however, with a careful design this cost remains proportional to $O(logN)$ (explained later in Section 4).

As the nodes are aware of the position of their zones, they can easily determine the most distant point in d dimensions. To do so, we made a simplification and used the bottom-left corners of a zone as reference points to define zone-to-point distances. The bottom-left corner is a good choice since the zone splitting rules of CAN keep this point always belonging to the original owner of the zone, no matter how many times the zone is split because of joins. Note that routing in CAN can only progress horizontally or vertically passing through neighbors, and thus the most distant point is determined accordingly.

Fig. 3 shows a two-dimensional ID space; for the sake of simplicity, first we explain the LR-CAN algorithm for $d = 2$. For node *A*, located in the middle of the unit square, the most distant points are located in the corners of the ID space, which correspond to one specific point, (0,0), as all the four corners of the square represent the same point on the d-torus. The Euclidean distance between them is $r_{max} = 2 \cdot \frac{1}{2}$, as we need to progress through half of the full length of the first dimension horizontally then again half of the full length of the second dimension vertically. To cover that distance, we will thus need

$$\widehat{R}_{max} = r_{max} \cdot N^{\frac{1}{2}} \quad (1)$$

hops, as along each dimension there are approximately $N^{\frac{1}{2}}$ nodes dividing the coordinate space among them [1]. If *A* initiates a lookup to (0,0), then it can measure the hop distance to the most distant point by a counter in the packet; based on the result and by transforming Equation 1, it can assess N ($\widehat{N}$). Then, we can calculate the value of $SR(\widehat{N})$, which is supposed to be the upper bound for the average route length. If the average lookup cost ($\widehat{R}_{avg}$) that can be calculated from the measured $\widehat{R}_{max}$ (explained in section 4) is higher than the value that the cost-limit function ($SR(N)$) allows, a new level of long-range neighbors is deployed; by doing so, we reduce the size of the ID space where lookups use traditional CAN routing, and we add the first level of long-range contacts, where long-range routing takes over ($L = 0$). Certainly, the distant point we measured our distance to belongs to a zone for which there is a responsible node. This node will be added as a long-range contact, and will be used as any other short-range neighbor in greedy forwarding. Network nodes behave consistently; thus, their individual decisions will be valid in a global scope, improving the performance of the whole network.

If all nodes assessed N and decided to set up their first long-range contact to the most-distant point compared to their own position, the traditional CAN routing will have to be employed only over half of the original ID space for every node; in this reduced space, the most distant points from the nodes will be half as far as the point where a long-range contact points already. For node *A*, the most distant

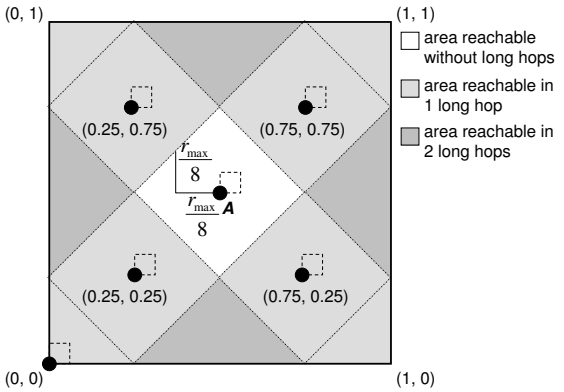

Figure 4. Two long-range levels are active

points (measured in hops) lie on the edges of the square spanned by points $(0.5, 0)$, $(1, 0.5)$, $(0.5, 1)$, and $(0, 0.5)$ (the white square in Fig. 3). If the system intends to further reduce the maximum distances, it needs to distribute the rest of the ID space evenly to have a general improvement that holds for the whole ID space. Among the most distant points, points $(0.25, 0.25)$, $(0.25, 0.75)$, $(0.75, 0.75)$ and $(0.75, 0.25)$ are the most appropriate choices for this goal, as shown in Fig. 4. These points are the next level long-range contact candidates for *A*. In case the hop distance to these new points exceeds $SR(\widehat{N})$ again—which is measured periodically—a new level of long-range contacts is introduced ($L = 1$). In this case we will initiate four new long-range connections; as we do not talk about the entire ID space anymore, these candidate points are not collocated on the torus. By introducing these new long-range contacts, the ID space where the original CAN routing is employed will be reduced again. The whole procedure is repeated until the average hop distance of the traditional CAN routing does not exceed the route length limit dictated by $SR(\widehat{N})$; hence, the LR-CAN algorithm endeavors to maintain the following condition:

$$\widehat{R}_{avg} = \widehat{R}_{max} \cdot m_{\frac{avg}{max}} < SR(\widehat{N}), \tag{2}$$

where $m_{\frac{avg}{max}}$ is a multiplication factor, an empirical ratio of R_{avg} and R_{max} that can be estimated accurately on-line based on the current number of long-range levels and $\widehat{R}_{max}$.

From now on we will call the traditional CAN routing as *short-range routing* because it uses only short-range contacts in forwarding; accordingly, we call *long-range routing* the case when long-range contacts forward a lookup. The routing of LR-CAN can be split in two phases: in the first one a certain key is approached by long-range routing; in the second phase short-range routing leads to the owner of the key. In Fig. 3 and Fig. 4 we can see how the deployment of long-range contacts affects LR-CAN routing. The white square shows the subspace where only short-range routing is employed, in case node *A* starts a query. If the query is initiated to some distant part of the ID space, *A* uses its long-range contacts first, and only afterwards the traditional CAN routing, based only on short-range contacts; thus, the scope of traditional CAN routing is reduced. Certainly, the $SR(N)$ cost-limit function limits the route lengths only partially since the addition of new long-range levels will generate a long-range routing cost.

An LR-CAN node needs to maintain its long-range contacts to review the assignment of coordinate points to nodes. To do so, it pings its contacts periodically and directly. If one of them fails, in the next maintenance period the node initiates a lookup for the corresponding coordinate point again, in order to find out which node is currently responsible for that point. The join procedure of CAN also needs to be modified. To spare the cost of setting up long-range levels, the newcomer can learn the current number of levels from the node through which it joins the network. Nodes also need to periodically route a message to one of the most distant points of the ID space, in order to measure whether a new long-range level has to be installed or not; this is done together with contact maintenance. Since long-range contacts are not necessarily symmetric, a node's long range contacts do not have to be notified neither when the node joins, nor when it leaves the network. As the bottom-left corner of a zone is invariant while a node is online (as explained by Fig. 2), if a coordinate of a long-range point hits the bottom-left corner of the given long-range contact, the point-to-node assignments do not change until the node leaves the network.

4. LR-CAN Cost Analysis

If we assume that routing can progress only horizontally or vertically, the maximum distance in one dimension between two points $\mathbf{p}$ and $\mathbf{q}$ in the ID space, ($\mathbf{p}, \mathbf{q} \in \{\mathbb{R}^d | [0, 1)\}$) can be $\frac{1}{2}$. In d dimensions this maximum distance is $r_{max} = \frac{d}{2}$ (see Fig. 3). In order to obtain the distance in hops, the maximum distance has to be multiplied with the "resolution" ($N^{\frac{1}{d}}$) of the ID space. Therefore, in the original CAN solution the maximum route length in hops is

$$R_{max} = \left\lceil r_{max} \cdot N^{\frac{1}{d}} \right\rceil \tag{3}$$

if we have N nodes in the network [1]. Each new level of long-range contacts halves the maximum distance. Hence the average route length of short-range routing (not including jumps on long-range contacts) is:

$$R_{avg} = \frac{r_{max}}{2^{L+1}} \cdot N^{\frac{1}{d}} \cdot m_{\frac{avg}{max}}. \tag{4}$$

Through simulations we observed that in general $m_{\frac{avg}{max}} \sim 1.4$ when $L \geq 0$, and ~ 2 if no long-range contacts are used

(this latter observation is also present in the original CAN paper [1]).

We want LR-CAN to provide $O(logN)$ lookup cost similarly to other DHT solutions; as a consequence, we propose to use $SR(N) = \frac{1}{c} \cdot log_2 N$ as the upper bound of short-range routing, where c is the cost-limit factor, an arbitrary positive real number. To express lookup cost, we need to deduce how L depends on N. Substituting R_{avg} by $\frac{1}{c} \cdot log_2 N$ in equation 4 yields to the following:

$$\begin{aligned} L &= \log_2 \frac{r_{max} N^{\frac{1}{d}} m_{\frac{avg}{max}} c}{2 \log_2 N} \\ &= \log_2 r_{max} N^{\frac{1}{d}} m_{\frac{avg}{max}} c - \log_2 2 \log_2 N \\ &= O(\log N) - O(\log \log N) = O(\log N) \end{aligned} \quad (5)$$

as d can be considered as constant in our algorithm. The average message cost of lookups can be expressed by the sum of the long-range and short-range routing cost:

$$LR_{avg} + R_{avg} = O(L) + O(\log N) = O(\log N) \quad (6)$$

As mentioned earlier in section 3, probing if R_{max} is over the cost-limit function requires to route a probe message to one of the current most distant points (depending on the number of activated levels) in the ID space by using original short-range routing. Generally, if LR-CAN keeps R_{avg} proportional to $O(logN)$ then R_{max} is also proportional to it. However, in a pathological case when the size of the network grows with orders of magnitude between two long-range maintenance periods, determining R_{max} will cost $O(dN^{\frac{1}{d}}/2^L)$.

As a result, the introduction of L eliminates the significance of d in the lookup cost of LR-CAN. Although in the traditional CAN architecture, as we mentioned earlier, d can be set so as to provide low lookup cost, the signaling load will increase much more than in case of LR-CAN with multiple levels of long-range neighbors. Moreover, d is a fixed network parameter the change of which needs costly rehashing of the whole ID space. It is important to note that the ability to build a multi-dimensional space still has significant benefits, since it vests the system with better fault-tolerance (more short-range contacts); moreover, the imprecision of long-range contacts is less detrimental for lookup performance than in a single-dimensional ID space.

The reduced lookup cost of LR-CAN comes at the price of having more contacts to maintain than in CAN:

$$\begin{aligned} LC_{avg} + SC_{avg} &= 2d + 1 + L2^d \\ &= O(d) + O(L2^d) \\ &= O(L) = O(\log N) \end{aligned} \quad (7)$$

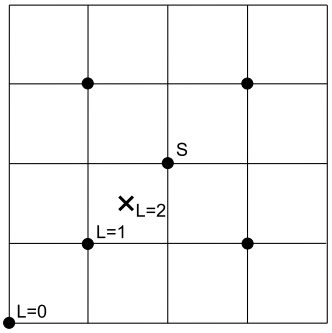

Figure 5. A level-two long-range coordinate maps to the same node as a level-one

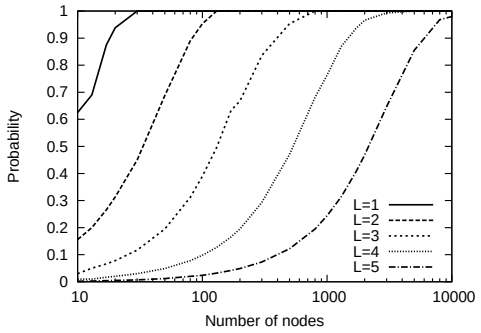

Figure 6. Probability of mapping all long-range coordinates to different nodes

as for level $L = 0$ there is 1 long-range contact, and for each level $L > 0$ there are another 2^d contacts.

Certainly, L has an implicit upper bound; its margin depends on the "resolution" of the ID space. There is no point in defining new long-range coordinates if they are so close to the given node that they map to the same nodes; this case can be seen in Fig. 5: the bottom-left level 2 contact of node A will map to the same node that holds its level 1 contact. Consequently, L should be defined on the $(-1 \leq L \leq \lfloor \log_2 N^{\frac{1}{d}}/2 \rfloor, L \in \mathbb{Z})$ domain; -1 indicates the case when no long-range contacts are used. Obviously, this constraint will be different in real life since the mathematical formula presented here assumes an equally distributed ID space. In Fig. 6, we can see the probability that all long-range coordinates of a given level map to different long-range contacts in function of network size. These probabilities were determined by simulation that used a uniform random number generator to draw the "join coordinates" of CAN nodes. For instance, for 256 nodes the theoretical upper bound yields maximum three levels, while we can be sure about mapping all contacts on the three levels to different nodes with a probability of around ~ 0.88, ac-

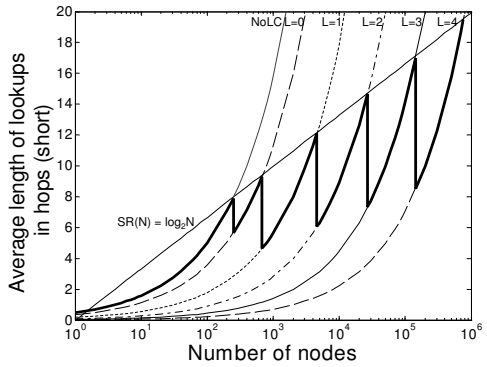

Figure 7. Analytical results on the growth of $\mathbf{R_{avg}}$ in function of $\mathbf{N}$

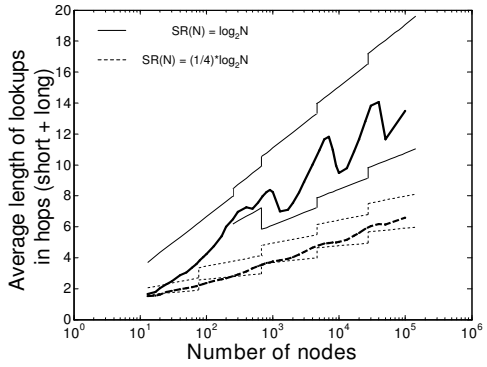

Figure 8. Simulation results on the average route length for different SR functions

cording to the simulation. As we learned, the desired cost-limit function is expressed through the c parameter, but the domain of L implicitly defines also the reasonable domain of c, i.e., there is no point in defining a too high c value. Nevertheless, the protocol presented in Section 3 can safely handle any arbitrarily high c and set up long-range contacts only when beneficial.

5. LR-CAN Performance Simulation

We simulated LR-CAN in a P2P overlay simulator [10, 11] developed in cooperation with the Technical University Darmstadt. The simulator can be used for testing several well known P2P search/lookup algorithms. Currently, Chord, Kademlia, CAN, and Gnutella [12] are all implemented in the simulator. We used the simulator to implement LR-CAN, validate our mathematical formulas in a simulated environment, and to compare the routing performance of CAN, LR-CAN, Chord, and Kademlia.

5.1. Validation of Theoretical Results

In Fig. 7, we can see how short-range route lengths grow in function of network size, in a two-dimensional CAN space, according to the above presented analytical model. A new level is switched on when the $f(N) = dN^{\frac{1}{d}}$ curves cross the desired average of short-range routing, the cost-limit function, namely $SR(N) = \frac{1}{c} \cdot log_2 N$. In Fig. 7, $c = 1$. When $SR(N)$ is reached, route length drops, while the number of neighbors increases. The number of levels corresponding to the different curves is denoted by L. The average, theoretical route length of the short-range routing grows along the thick line.

The mathematical analysis contains assumptions that cannot be fully met in simulations and real life. For the simulation results, we separately constructed LR-CAN networks with several different network sizes, and performed a significant amount of random key lookups to obtain a representative average for lookup message cost. In Fig. 8, we can see how LR-CAN with different c parameters reacts on a growing network. We can see the simulated results on the thick curves with two different values of c ($c_1 = 1$ and $c_2 = 4$) to define two different cost-limit functions. To compare the simulated results to the analytical ones, the thin lines represent a lower and an upper bound for the simulation results, which enclose the respective thick curve. The jumps in the thin curves are due to the long-range routing cost (LR_{avg}) that appears when a new long-range level is introduced; this routing cost depends only on the value of L and d (see Section 6). The theoretical thin curves assume equal-size, 2-dimensional zones and the same L value for all nodes, assumptions which obviously do not hold in real setups.

5.2. LR-CAN Comparison with Chord and Kademlia

Having multiple dimensions is a solution to speed up lookups. Linear search in a one-dimensional space generates $O(N)$ lookup lengths, compared to the $O(dN^{1/d})$ lengths that CAN provides in a d-dimensional space; higher d values yield lower lookup cost. Consequently, when a multi-dimensional solution, such as LR-CAN, has to provide a determined lookup cost, it may have less strict requirements for the precision of long-range contacts than in single-dimension solutions, such as Chord and Kademlia. As a result, LR-CAN may generate less signaling for the maintenance of long-range contacts.

Another DHT solution, Chord, is similar to LR-CAN except for three essential features. First, Chord uses a one-dimensional ID space, which requires to pay more attention to the precision of long-range contacts in order to keep

lookups fast. Second, the ID space is unidirectional, so less alternative routes exist in the system (this drawback was overcome by [3]). And third, the distribution of the ID space is more heavily based on randomness, i.e., the position and hash zone of a node on the ring basically depends on the ID of the node. As opposed to this, CAN splits the ID space into equal parts as much as possible, according to its zone splitting rules operating regardless of the node IDs. A CAN ID space can be easily mapped to a nearly balanced binary tree. Thanks to these features, the peers participating in an LR-CAN network can estimate individually the actual global average of lookup cost, and decide about the introduction of new levels of long-range neighbors accordingly. This is a major advantage of LR-CAN.

Kademlia nodes cover each part of the ID space with k randomly selected neighbors stored in a so called "k-bucket." Kademlia significantly differs from CAN and Chord as it is an on-demand, reactive solution, while the latter ones are proactive. The on-demand nature origins from the contact maintenance procedure. While CAN and Chord are proactive in short-range contact maintenance, Kademlia learns topology changes from its own lookup traffic. If there is no traffic at all, or just moderate one, Kademlia may have imprecise information about the network topology, and thus, it may fail to answer certain lookups or move key-value pairs to the appropriate nodes. As a consequence, Kademlia needs to make some effort to spread the topology information better. To do so, Kademlia nodes generate dummy lookups periodically, and employ an iterative lookup strategy. This means that a lookup message is always returned to its initiator after every newly visited hop, until the destination is found; this results in higher costs for the lookups. Another way to better disseminate topology information is to increase the k parameter. In this case, a node learns about more nodes in the network, and thus knows the topology better; resources are replicated and stored on more nodes, so they are "easier" to find.

In our simulations we focused on performing a *fair comparison* of these DHTs. The scope of the simulations was to investigate how dynamism affects the message cost of lookups, which is basically influenced by the precision of long-range contacts. Since the effect of failures and graceful leaving of nodes are similar in terms of long-range contact maintenance, we did not implement failure handling mechanisms in Chord and LR-CAN. As the on-demand nature of Kademlia enables to handle failures without any additional mechanisms, the comparison in Fig. 9 is only partly relevant for Kademlia because, as mentioned, Chord and LR-CAN lack the corresponding mechanisms. The investigation of failure-tolerance can be another important topic of DHT-related research [13]. The elimination of failure handling means that failure detection and its corresponding mechanisms are not implemented; nodes always "clean up" after they leave the system, i.e., short-range contacts remain correct and precise, and stored key-value pairs are moved to the node that takes over the responsibility of handling the leaving node's zone. By these means, we endeavored to simplify the maze that this comparison with its multi-dimensional problem space frames. We can clearly focus on how the precision of long-range contacts and the stabilization of the DHTs maintaining these contacts affect average lookup costs and the signaling DHTs generate.

Parameters. One of the most influencing parameters is the length of the *stabilization period* DHTs have. All of these three DHTs operate some procedure with a common aim: to set the precision of long-range contacts. The name "stabilization" origins from the procedure of Chord that involves into stabilization a so called "fix-fingers" procedure. As described in Section 3, LR-CAN also reviews its long-range contacts periodically. In Kademlia, a similar periodic task is fulfilled by bucket refreshes. In order to measure how tolerant the algorithm and the overlay structure of these DHTs are for the imprecision of long-range contacts, we varied the stabilization period on a wide range.

LR-CAN and Kademlia have other system parameters that highly affect their performance. As mentioned earlier, the number of dimensions is still an optional parameter that can vest LR-CAN with better failure tolerance. Additionally, dimensions affect when and how long-range contacts need to be deployed. The effect of c has already been deeply discussed; it represents the connection between network size and average lookup cost. Kademlia also has two important parameters: the value k has influence on how widely routing information is disseminated and how redundant the system becomes. Furthermore, parameter α is a concurrency parameter defining the number of asynchronous parallel queries a node can start on a lookup request. We omitted the investigation of α in this comparison as this type of concurrency primarily exploits the underlying link characteristics and could be implemented both in Chord and LR-CAN.

Metrics. A usual way to evaluate the performance of different DHTs is to compare the average number of logical hops needed to find the owner of a key. However, there are other metrics that might come also into focus when logical hops are compared. First, there is a difference between recursive and iterative protocols, as they approach a target differently. A lookup flowing through the same number of nodes means twice as many messages for an iterative protocol (e.g., Kademlia), as the initiator controls the lookup process hop-by-hop. As a result, the *message cost of a lookup* is a better metric than the number of logical hops. Note that in the measurements related to message cost we counted the number of messages needed to reach the owner of a key; the last message, used for returning the searched value, was not added. The amount of *signaling traffic* together with

the achievable message cost of lookups determine the overall lookup performance. Low message cost is worth nothing if its provision requires relatively high signaling traffic. In our signaling traffic measurements we included all the traffic generated during node joins and leaves, the stabilization of long-range contacts, and the lookup traffic. If the intention is to compare the results with Kademlia as well, lookup traffic also needs to be involved into signaling, since Kademlia uses its lookup traffic to maintain its long-range contacts; thus, contact maintenance cannot be differentiated from lookup traffic as in Chord and LR-CAN.

There are also other metrics that might be interesting, but we omitted to present them as they either seemed redundant with the above mentioned metrics, or were non-informative in our simulations. One of these metrics was the *number of contacts* that is only loosely proportional to signaling traffic. For instance, Kademlia may store several times more contacts than Chord or LR-CAN but it generates far less signaling messages. Average *latency of lookups* was a redundant metric in our simulations as non of the implemented DHTs was optimized for link latency; thus, on long-run average the message cost of lookups was directly proportional to the average lookup latency. Nevertheless, the *success ratio of lookups*, which reflects the ratio of the number of successful lookups divided by the number of lookups for a stored key, is an important metric in failure scenarios. However, as already mentioned, in our simulations nodes always cleaned up after leaving; hence, in a proper implementation of Chord and LR-CAN the success ratio of lookups remains 100%. In Kademlia this is not so straightforward: for low k values or rare stabilization steps a node trying to push its key-value pairs to the appropriate nodes before leaving may not be properly aware of the nodes closest to the given keys. This may result in loosing some of its keys and thus the network gradually looses a portion of the key-value pairs it should have saved. This phenomenon can be effectively reduced with increasing k or the stabilization period, which will implicitly also appear in some of the message cost figures we present.

Scenario. As the aim of the simulation was to test how the imprecision of long-range contacts influences routing in the overlays of the different DHT algorithms, we labored a scenario where we replace a significant portion of the participating nodes several times to generate stale or imprecise long-range entries. For the sake of fair comparison between different phases of the simulation run, we split the simulation time up to five phases that are equal in time (1470 sec). In the first phase, 10000 nodes gradually join the network and initiate the storage of some random key-value pairs. These contents are also stored in a globally accessible memory in the simulator so that later the nodes can perform lookups only for content that already had been stored and that should have been preserved by the DHT during the simulation. Note that in the simulator implementation a key stored in the DHTs is not removed when its original owner leaves the network; hence, in an ideal case, DHTs should preserve all the keys that were stored sometime during simulation. Performing lookups for stored content is important especially for Kademlia since in this DHT lookups for a key without a responsible node usually roam around longer than lookups for keys with a responsible node. In the second phase of the simulation, each node performs 10 lookups for different random stored keys. In each of the remaining parts, 32% percent of the nodes is replaced in a 400-sec churn window. In these windows, nodes first leave and then new nodes gradually join the system; the network population always resets to 10000 nodes, so as to preserve the fair lookup and signaling comparison between the phases. Nodes that join the system run the stabilization procedure and fix their long-range contacts according to the current network topology. The next stabilization is called when the timer initiated with the stabilization period elapses; hence, the stabilization periods of nodes are asynchronous, and depend on the time they joined the network.

Analysis of Results. For each simulation run we present two figures. One figure shows the evolution of lookup message cost over time. In order to reduce the number of points, each point represents the average of 1000 consecutive lookups. In order to estimate the value of a result we need to see also the price at which it was obtained. Thus, beside lookup message cost we also present the absolute number of signaling messages that enabled the corresponding performance. One box in these figures depicts the messages sent only in the given simulation phase. Since there are many parameter set combinations, for figures in Fig. 9, we selected a parameter set that adequately represents the protocols, and we altered only the stabilization periods.

In Fig. 9 we can observe several interesting results and phenomena. The most striking one is that the lookup message cost of LR-CAN with the current parameter set is consistently lower in almost all cases. Certainly, with different d and c parameters, we can experience different performance; with higher c value, LR-CAN can further lower lookup message cost. With frequent stabilizations, dynamism has almost no effect on lookups in Chord and LR-CAN; these DHTs keep the same lookup performance over time. In contrast, Kademlia has an interesting behavior on churn. In the second phase, Kademlia has an outstanding lookup cost that deteriorates rapidly in the third phase after the first churn window, which results in many stale entries. After a while, lookup cost converges to the performance experienced before the churn window, but this takes a long time. In the next phase, we can observe a slight shift upward in lookup cost because Kademlia looses some of the keys stored at the beginning of the simulation due to churn and stale entries, and lookups for these keys last longer.

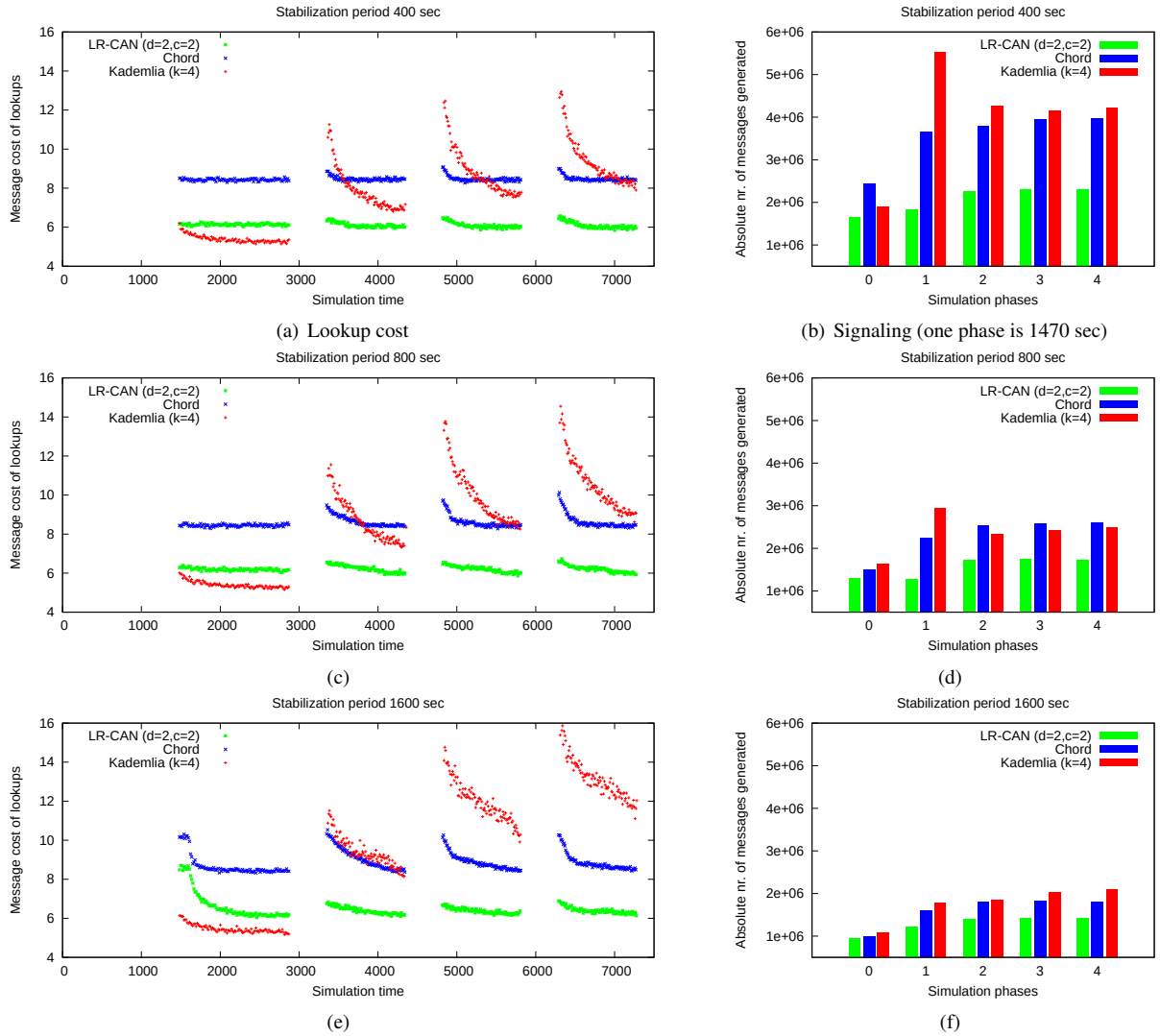

Figure 9. Lookup performance and corresponding signaling traffic in a dynamic network

In Fig. 9(b) we can see the corresponding signaling traffic that contains also the lookup message cost presented by Fig. 9(a). Note again that Kademlia uses its lookup traffic to maintain long-range contacts as well, so maintenance could not be separated from lookups. LR-CAN needs less than half of the signaling of Chord and Kademlia that partly comes from the lower lookup message cost and the more efficient repair of stale long-range entries. In phase 1 we can observe a slightly higher signaling traffic for Kademlia than in later phases, since in this phase there are constantly 10000 nodes present; thus, more bucket refreshes are performed, which are more costly than Chord and LR-CAN stabilizations.

In the rest of the lookup cost figures we can observe that the vertical shift of Kademlia over time becomes more significant, and its ability to converge to the original lookup cost average reduces. Both for Chord and LR-CAN some time is needed until long-range contacts become accurate. With longer stabilization periods, this convergence time also increases. Nevertheless, with its one-dimensional and unidirectional ID space, Chord is more sensitive to network dynamicity; thus, the average lookup cost deteriorates more than in LR-CAN. Regarding the signaling load, the difference between the protocols gets less significant; however, LR-CAN keeps its preliminary advantage.

6. LR-CAN Signaling Optimization

So far the aim of this work was to minimize the message cost of lookups, assuming a reasonable signaling traffic. As

we have seen, the amount of signaling traffic gives a clear view on how much the provided lookup cost is worth. We have also seen that with a well selected stabilization period we may loose some negligible advantage in lookup cost, but we can spare significant signaling traffic load. Besides the stabilization period, the other influential parameter on signaling in LR-CAN is the parameter c that drives the number of long-range levels in a network. Since in a DHT a high portion of the signaling traffic origins from the maintenance of long-range contacts, there might be cases when this kind of network "investment" is not equilibrated by the gains in lookup cost (which is another type of signaling traffic). The low number of lookups injected in the system results in some long-range contacts becoming superfluous; their maintenance costs more than what can be gained on their availability. Overall, fast lookups are desirable in a DHT, but their provision requires a not negligible amount of signaling traffic. Beside the provision of fast lookups, our second aim could be to minimize thus the signaling traffic, i.e., lookup traffic and signaling related to contact maintenance. The rest of this Section describes how most of the signaling traffic can be characterized in LR-CAN by other parameters, and presents a method (together with its specific assumptions and conditions) through which we are able to optimize the number of long-range levels L and minimize signaling traffic.

In LR-CAN, most of the signaling traffic can be well described with some formulas; hence, we can optimize the value of the input parameters in order to minimize signaling traffic accurately. With a few assumptions and modifications to the original LR-CAN algorithm, we can construct an algorithm (SIGMIN) that adapts the number of long-range contacts not only to network size (N) but to lookup rate (λ_l) as well. Here we exclude the cost-limit function from the optimization procedure since in this task we do not care about lookup cost, only about the minimization of aggregated signaling traffic.

The problem is thus to find an optimal value for the number of long-range levels (L) in function of network size (N), long-range maintenance frequency (λ_p), and lookup rate (λ_l). A method for gathering information about N was described earlier in Section 3. The frequency of long-range maintenance, i.e., the stabilization period λ_p is a global network parameter. However, for λ_l we need to implement an aggregated-data-collection technique which collects lookup rate data from the entire network, or at least from a significant part of it. In our solution, data collection is initiated by the bottom-leftmost node of the ID space, which can be a dedicated anchor node in the LR-CAN if it is persistent since the launch of the system; the collection has a cost of $O(N)$. After the data collection, this node can evaluate the input parameters and find the optimal value of L, which is then broadcast to the network. Consequently, the earlier definition of L, which was presented in Section 3 and was based on individual decisions of peers, has to be changed to a method where one node derives the optimal value of L based on aggregated, global information.

By the following equations, we look for the L value that yields the minimal aggregated cost of long-range contact maintenance ($M_{cost}(L)$) and lookup cost ($L_{cost}(L)$) for the whole network, given the parameters described above. $M_{cost}(L)$ includes the message cost of checking one long-range contact, the network size N, the maintenance frequency (λ_p), and the number of long-range contacts when L levels and d dimensions are used (see Equation 7). $L_{cost}(L)$ includes the network size, the lookup rate (λ_l), and the routing cost on long- and short-range contacts (see Section 4).

$$M_{cost}(L) = \begin{cases} 2 \cdot N \cdot \lambda_p \cdot (1+L), & \text{if } -1 \leq L < 0 \\ 2 \cdot N \cdot \lambda_p \cdot (1+L \cdot 2^d), & \text{if } L \geq 0 \end{cases} \quad (8)$$

$$L_{cost}(L) = N \cdot \lambda_l \cdot (LR_{avg}(L) + \frac{\frac{d}{2} N^{\frac{1}{d}}}{2^{L+1}} \cdot m_{\frac{avg}{max}}), \text{if } L \geq -1 \quad (9)$$

$$LR_{avg}(L) \approx \begin{cases} 0.5 \cdot (1+L), & \text{if } -1 \leq L < 0 \\ 0.5 + d \cdot 0.343 \cdot L, & \text{if } L \geq 0, d = 1, 2 \end{cases} \quad (10)$$

The global optimum for L is:

$$L_{opt} \approx \arg \min_{\left(\left[-1, \log_2 N^{\frac{1}{d}}/2\right], L \in \mathbb{R}\right)} (M_{cost}(L) + L_{cost}(L)) \quad (11)$$

For the long-range cost $LR_{avg}(L)$, we gave a close approximation in equation 10, derived from the average cost resulting from the following formulas:

$$\begin{aligned} c(L) =& 2^L \cdot \left(\frac{1}{2} + \frac{L}{2}\right) - \\ & -2^L \cdot \left(\sum_{i=2}^{L} \left(\frac{1}{2^i} + \sum_{j=1}^{\lfloor \log_2 (i-1) \rfloor} \frac{1}{8^j}\right)\right), \text{if } L \geq 1 \end{aligned} \quad (12)$$

$$LR_{avg}(L) = \begin{cases} \frac{1}{2}, & \text{if } L = 0 \\ \frac{(c(L)-1)}{2^L} + \frac{1}{2^{L+1}}, & \text{if } d = 1, L \geq 1 \\ \frac{(c(L)-1)}{2^{L-1}} + \frac{1}{2^{2L+1}} - & \\ -4 \cdot \sum_{i=1}^{\lfloor \log_2 (L+1) \rfloor} \frac{1}{4^i} \cdot \sum_{i=1}^{\lfloor \log_2 (L) \rfloor} \frac{1}{4^i} & \text{if } d = 2, L \geq 1 \end{cases} \quad (13)$$

The validity of these formulas can be checked against the weighted average of routing zones, as presented in Fig. 3 and Fig. 4. Equations 12 and 13 prove the validity of equation 10 only for maximum two dimensions; however, two dimensions are adequate for LR-CAN and SIGMIN, since the lookup performance does not depend on d in these two algorithms, as mentioned in Section 4.

Solving equation 11 yields the optimal value of L, which will be a real value. However, on a given node, L must have

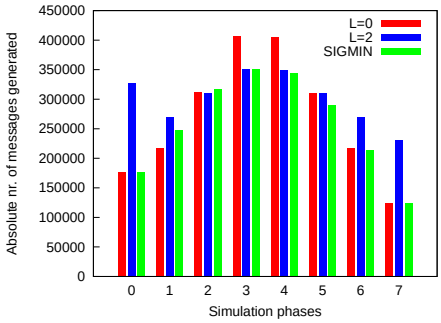

Figure 10. Simulation of lookup rate influence on signaling

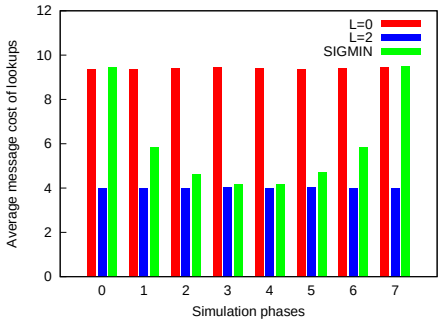

Figure 11. Corresponding lookup cost of the same simulation

an integer value. Thus, the nodes will individually choose the integer value either immediately below or immediately above the optimal value, using a simple stochastic method. As a result, the global average of L will be close to the optimal value.

We need to optimize the performance of the network in the future based on past data. Hence, peers should provide detailed information about their lookup activity on a regular basis (e.g., daily), not only a simple average of a certain larger timeframe. On the other hand, the decision made in advance is only valid if network population does not decrease or increase significantly in the timeframe the decision is made for, or the change has to be predictable.

The simulation that produced the graphs in Fig. 10 consisted of equal timeframes with constant node population. Only lookup rate changed in each phase, linearly increasing from phase 0 to 3, being constant in phase 3 to 4, and decreasing from phase 4 to 7. In one case L was set to 0, in another case L was set to 2 in a static manner for the entire network, while in the third case nodes used the SIGMIN algorithm to determine the value of L. SIGMIN was started with no long-range contacts. Note that setting up new levels of long-range contacts has some non-negligible cost, which is clearly shown in the first phase for the $L = 2$ variant (boxes in the middle); if $L = 0$, there are much less contacts to set up and maintain (boxes on the left). SIGMIN (boxes on the right) might also be affected by this additional signaling cost, as an increasing lookup rate might trigger the introduction of new long-range contacts, according to equations 8–11. This effect can be seen in phases 1 and 2; hence, in phase 2 SIGMIN generates a higher signaling load than the other solutions, although only by a slight margin (this casual cost is not expressed by the formulas 8–11). The remarkable ability of the SIGMIN algorithm is to always converge to the more efficient variant in terms of signaling (Fig. 10). The average message cost of lookups corresponding to the variants is depicted in Fig. 11.

7. Conclusion

In this paper, we proposed an enhanced algorithm for CAN, which reduces the scope of the original routing method in the ID space by deploying long-range contacts. These contacts are deployed adaptively, so as to limit the maximum possible route length in the original greedy forwarding step of CAN. If the lookup cost exceeds a defined limit as network grows, LR-CAN deploys new levels of long-range contacts. As a consequence, the lookup cost becomes proportional to $O(logN)$ for the right $SR(N)$. Our simulations prove our theoretical performance evaluation to be appropriate. Compared to popular DHT algorithms (e.g., Chord and Kademlia), LR-CAN has an outstanding performance in networks of large-scale and even in dynamic scenarios.

We also presented a method to describe the signaling traffic of LR-CAN originating from message cost of lookups and contact maintenance. By a few measurable parameters we can optimize the number of long-range contacts in the network to minimize the signaling traffic of LR-CAN.

Acknowledgment

The authors would like to the Hungarian National Information Infrastructure Development Programme for the supercomputer cluster we were allowed to use for our time and resource consuming simulations.

References

[1] S. Ratnasamy et al. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM*, pages 161–172, San Diego, CA, 2001.

[2] I. Stoica et al. Chord: Scalable peer-to-peer lookup service for internet applications. In *Proc. of the ACM SIGCOMM*, pages 149–160, San Diego, CA, 2001.

[3] J. Jiang et al. Bichord: An improved approach for lookup routing in chord. *Lecture Notes in Computer Science, ADBIS*, 3631:338–348, 2005.

[4] P. Maymounkov and D. Maziéres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proc. of the 1st IPTPS'02*, pages 53–65, Cambridge, MA, 2002.

[5] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350, Heidelberg, Germany, 2001.

[6] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[7] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing*, pages 163–170, Portland, OR, 2000.

[8] A. Ohtsubo et al. The power of shortcuts in greedy routing in content-addressable networks. In *Lecture Notes in Computer Science, EUC*, volume 3207, pages 994–1003. Springer Berlin, 2004.

[9] S. Ratnasamy et al. A scalable content-addressable network. In *TR–00–010*, Berkeley, CA, 2001.

[10] V. Darlagiannis, A. Mauthe, N. Liebau, and R. Steinmetz. An adaptable, role-based simulator for p2p networks. In *Proc. of Int. Conference on Modeling, Simulation, and Visualization Methods*, pages 52–59, Las Vegas, NV, 2004.

[11] Peerfactsim.kom. http://www.peerfact.org, 2007.

[12] Gnutella. http://www.gnu.org, 2007.

[13] D. Liben-Nowell et al. Analysis of the evolution of p2p systems. pages 233–242, 2002.

[14] B. Kovács and R. Vida. An Adaptive Approach to Enhance the Performance of Content-Addressable Networks. In *Proc. of ICNS 2007*, pages 93–98, Athens, Greece, 2007.