Cyclic Routing: Generalizing Look-ahead in Peer-to-Peer Networks

Dmitry Korzun*
Department of Computer Science,
Petrozavodsk State University (PetrSU)
Petrozavodsk, Republic of Karelia, Russia
Email: dkorzun@cs.karelia.ru

Boris Nechaev and Andrei Gurtov
Helsinki Institute for Information Technology (HIIT)
P.O. Box 9800, FIN-02015 TKK, Finland
Emails: boris.nechaev@hiit.fi, gurtov@hiit.fi
Fax: +358 9 694 9768

***Abstract*—Distributed Hash Tables (DHT) provide a lookup service in peer-to-peer overlay networks. Many valuable applications have been recently built on top of several available DHTs. However, they function poorly when no direct IP connectivity is available to some nodes (e.g., located behind a NAT or firewall) or in the presence of overloaded or malicious nodes. In this paper, we propose a new method for DHT-based routing called cyclic routing. It generalizes existing single-hop look-ahead approach (also known as "Know thy neighbor's neighbor") and supports multipath routing. The method provides a systematic way for collecting stable and efficient overlay paths. Cyclic routing has the same theoretical dependability and efficiency upper bounds as basic DHT routing but it is more resilient when IP connectivity is limited or when the overlay suffers from overloaded nodes.**

Keywords: Distributed hash tables, Peer-to-Peer routing, Lookup availability

I. Introduction

A Distributed Hash Table (DHT) provides a lookup of identifiers and routing to the node storing a corresponding value. Each node keeps contacts (node IDs and IP addresses in the local routing table) to some other nodes (neighbors), forming an overlay on the underlying network. For routing a packet, nodes sequentially forward it using communication primitives of the underlying network.

The IP-level reachability applies certain restrictions on DHT functioning. First, a pure DHT assumes a node may access an IP address of another node. In an untrusted environment, however, a node can prefer not to give its IP address to untrusted nodes to avoid direct attacks. Only trustworthy nodes may contact it directly via IP; communication with other nodes uses overlay node IDs.

Second, IP addresses are less stable than node IDs. For instance, when a node is mobile it can change its IP address frequently. Consequently, many routing table entries can become invalid, and high maintenance cost is required to provide routing tables up to date. In contrast, IP address changes should not necessarily lead to changes in P2P overlay paths.

Third, some nodes have no global IP addresses or a network can exhibit non-transitivity [1]. A node u cannot communicate directly with a node v, and an intermediate node w is needed, e.g., when u and v are separated by a NAT.

*Dmitry Korzun is a visiting research scientist at HIIT.

For a DHT it means that a node cannot add neighbors freely. Consequently, some routing improvements based on enhancing node knowledge about the overlay become less attractive (e.g., proximity neighbor selection [2] or routing tables of variable size [3]). Iterative routing becomes troublesome since a source node cannot always contact directly each node in a route. As a result, DHT routing suffers from IP-level reachability restrictions.

In this paper, we propose a new method for overlay routing on top of a DHT. We call the method *cyclic routing* (CR) since it uses cyclic paths (cycles). It generalizes existing ideas of single-hop look-ahead also known as "Know thy neighbor's neighbor" [4] and of redundant multipath routing [5].

Each node maintains a collection of cycles additionally to its routing table. A cycle is a path that starts from the source node to its neighbor, then runs through the overlay and returns to the source. Only node IDs are stored to identify a cycle. Cycles present global knowledge about paths in the overlay. In routing, a node can alternatively use either cycles or the underlying DHT. If there is a cycle with a node close to the destination, the packet is sent along the cycle. Otherwise, the underlying DHT selects the next hop. For illustration, we use the Chord DHT [6].

In fact, cyclic routing provides a systematic way for collecting stable and efficient overlay paths. It does not require knowledge of additional IP addresses nor conflict with routing improvements that enhance node knowledge about the overlay using IP addresses. In contrast, our method complements these proposals when they fail because of the IP-level reachability restrictions.

When IP-level reachability is restricted, CR is more resilient but preserves provable dependability and efficiency upper bounds of basic DHT routing. According to our simulations, CR also provides more resilient routing in the presence of malicious nodes.

The rest of the paper is organized as follows. Section II summarizes P2P routing and discusses existing proposals that allow nodes to learn more about the global network. Section III introduces the cyclic routing method. We analyze its properties and discuss the key design principles. Section IV presents simulations results evaluating lookup success rate of basic and CR-Chord under churn. In Section V, we discuss additional

978-1-4244-3806-8/09/$25.00 © 2009 IEEE

capabilities provided by cyclic routing, including multi-path and secure routing. Section VI concludes the paper.

II. Problem Domain

Consider a DHT-based overlay of N nodes. Let node IDs be assigned from an identifier space with a distance metric ρ. In a pure DHT, distance metric calculation is based on node IDs only. If $\rho(u,d) > \rho(v,d)$ then v is closer to d or v is in between u and d. Let n_{ud} be the number of nodes in between u and d.

Each node s maintains a local routing table T_s of entries (u, IP_u), where u is a neighbor and IP_u is its IP address. In a dynamic environment, nodes collaborate and adopt their routing tables to an up-to-date state. The number of neighbors $|T_s|$ is the node degree. Typically $|T_s| \ll N$, meaning that routing uses *local knowledge* about the overlay network.

In basic DHT routing, s forwards packets to u via the underlying IP network. A packet to a destination d goes sequentially through nodes whose IDs are progressively closer to d according to the distance metric. If $v \in T_s$ then s can forward a packet to v forming the one-hop path $s \to v$. Consequently, a multi-hop P2P path $s \to^+ d$ is constructed.

In the ideal case, DHT routing is dependable (a packet eventually reaches its destination) and efficient (a typical upper bound is $O(\log N)$ hops). In real environments, however, P2P paths can be long, inefficient for the underlying IP network, faulty, and insecure. Recently, a few methods have been proposed to improve efficiency and security of DHT routing.

Look-ahead. Many DHTs use greedy routing when nodes forward a packet to the neighbor closest to d. More efficient DHT routing techniques (e.g. [7]) do exist. Nevertheless, in previous proposals a node has little idea on the remaining route. An alternate approach exploits knowledge of neighbor's neighbor (one level of look-ahead) [4] producing shorter paths compared to greedy routing [8].

Flexible routing table maintenance. To reduce the route latency, s selects neighbors using knowledge of the underlying IP network. In particular, neighbors are chosen based on a proximity metric (proximity neighbor selection); selection of next hops depends not only on the ID distance but also on the geographical closeness (proximity route selection) [2], [9]. Moreover, nodes can increase the number of routing table entries. In this case, s varies $|T_s|$ independently on other nodes. When there is enough local resources, s inserts a new neighbor. When a neighbor is likely to have failed or to be malicious, it is removed. Proximity information, random sampling, and behavioral statistics are actively used [3], [10], [11].

Multipath routing. Having many neighbors, s forwards a packet via several alternate paths (using several neighbors), either in parallel or sequentially. This redundancy allows routing around failed or malicious nodes [5], [11], [12].

These methods use *additional knowledge* about the network. Nevertheless, in the first method, only a small step is done to analyze the remaining route; the last two require IP addresses, facing the problems of IP reachability restrictions.

Below we propose a routing method that generalizes look-ahead, supports flexible routing tables and multipath routing, but works with node IDs only. It complements the above IP-based proposals when they fail or become inefficient.

III. The Cyclic Routing Algorithm

Cycles are natural for bidirectional P2P communications. Let a node s send a packet to d; the path is $s \to^+ d$. Then d replies to s; the path is $d \to^+ s$. These paths form the cycle $s \to^+ d \to^+ s$. Taking intermediate nodes, $c = \big(s \to v_1 \to^+ \cdots \to^+ v_{l-1} \to^+ s\big)$, where $(v_1, \mathrm{IP}_{v_1}) \in T_s$ (direct IP contact). Nodes $v_2, \ldots, v_{l-1}$ may represent some but not all the nodes visited. It leads to the notation "$\to^+$" instead of "$\to$" in c.

Such cycles present possible routes in the overlay, generalizing the one-level look-ahead [4], [8]. Given a cycle c, a node s may assume that a packet sent to v_1 would go through $v_2, \ldots, v_{l-1}$. That is, s looks ahead of its neighbors.

A cycle uses only node IDs, except v_1 whose IP address is already in the routing table. Therefore, even if a node u cannot use IP_v, the node v can serve as an intermediate node being presented in a cycle stored at u. This is a key to deal with IP-level reachability restrictions.

Let a node u maintain (additionally to T_u) a collection of q cycles, $\mathcal{C}_u = \{c_j\}_{j=1}^{q}$, where $c_j = (u \to v_{j1} \to^+ \cdots \to^+ v_{j,l-1} \to^+ u)$. We call $\mathcal{C}_u$ a network *cyclic structure* known to u. It represents additional knowledge about the network.

The idea of cyclic routing (CR) is to find the next hop analyzing the remaining route. The latter is constructed at a current node using the cyclic structure. When no appropriate cycle exists, the underlying DHT is used, see Algorithm 1.

Algorithm 1 Cyclic Routing (CR)

Require: Packet p (traveling from s to d) arrives to $u \neq d$.
The node u maintains T_u and $\mathcal{C}_u$.

Find $c \in \mathcal{C}_u$ such that
$c = \big(u \to v_1 \to^+ \cdots \to^+ v_i = d' \to^+ \cdots \to^+ v_{l-1} \to^+ u\big)$
where d' is close to d;
if c is found **then**
Let v_1 be the next-hop node v;
else
Find the next-hop node $v \in T_u$ according to the underlying DHT;
end if
Forward p to v;

A. Cycle Transitions

Let a source node s select a cycle $c = \big(s \to^+ u \to^+ d \to^+ s\big)$. In Fig. 1(a), every intermediate node u uses the same cycle. The nodes, however, may be less coordinated. In Fig. 1(b), u selects a cycle $c' = \big(u \to v \to^+ w \to^+ d \to^+ u\big)$ such that its path $u \to^+ d$ does not coincide with the path $u \to^+ d$ of c.

In fact, the remaining route in CR is approximate. A packet follows a cycle, then transits to another, later it changes the

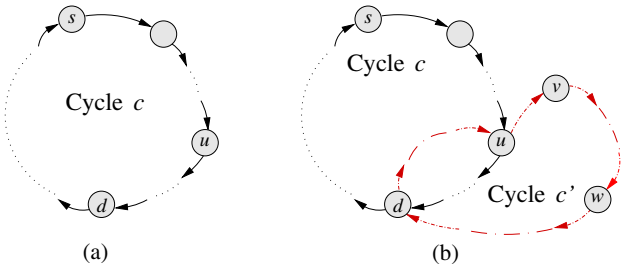

Fig. 1. Cyclic routing $s \rightarrow^+ d$: (a) one-cycle routing; (b) two-cycle routing

cycle again and so forth. One cycle is enough to route a packet, but there are transitions due to lack of coordination between nodes, absence of an appropriate cycle, or failures when a cycle does not reflect correctly the current network topology.

A node can fix a cycle when sending a packet. In this case, cycle node IDs are piggybacked to the data packet. The next-hop node tries to follow this cycle when possible.

B. Dependable Routing and Path Length Upper Bounds

In this section we consider two problems, (1) the CR dependability since cycle transitions make it questionable and (2) the CR path length compared with basic DHT routing.

We follow Castro *et al.* [13] in definition of the routing dependability. P2P routing is dependable if a packet sent from s to d is always delivered to d in a stable and correct network. In general, Algorithm 1 is not dependable. In Fig. 1(b), if w selects a cycle to reach d via u, then looping happens.

A dependable variant of the CR method exploits the same idea as in many DHTs—progressive routing. Any node forwards a packet progressively closer to the destination according to a distance metric ρ.

Theorem 1: Algorithm 1 is dependable if any next-hop node v satisfies

$$\rho(v, d) < \rho(u, d). \tag{1}$$

The number of hops is at most $\rho(s, d)/\delta$ where $\delta = \min_{u,v}(\rho(u, d) - \rho(v, d))$.

Proof: Inequation (1) states that v is closer to the destination, regardless of whether a cycle or the underlying DHT is used. Therefore, the distance to the destination decreases monotonically, and after a finite number of hops (not more $\rho(s, d)/\delta$) the packet arrives to d. ■

Some DHTs provide more efficient paths of length $O(\log N)$ using geometrical progressive routing when at every hop the distance is reduced by a constant $k > 1$. For instance it happens in greedy routing when a node selects the next hop as close as possible to the destination.

Theorem 2: Algorithm 1 is dependable if for some constant $k > 1$ any next-hop node v satisfies

$$\rho(v, d) \le \frac{\rho(u, d)}{k}. \tag{2}$$

There are $O(\log_k N)$ hops if n_{uv} is proportional to $\rho(u, v)$.

Proof: Inequality (2) is a particular instance of (1), thus the dependability is due to Theorem 1.

Let the route be $s \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{l-1} \rightarrow d$, where l is the hop number. Initially $n_{sd} < N$. Then $\rho(v_1, d) \le \rho(s, d)/k$, and the proportional reduction is $n_{v_1 d} < N/k$. At the next hop, $\rho(v_2, d) \le \rho(v_1, d)/k \le \rho(s, d)/k^2$, and $n_{v_2 d} < N/k^2$. Finally, it yields $N/k^{l-1} = 1$, and $l = O(\log_k N)$. ■

As an example consider the Chord DHT. The distance $\rho(u, w)$ is length (the number of all possible IDs) of the ring arc between u and w clockwise. A neighbor of u is a finger or successor. The ith finger is the closest node v such that $\rho(u, v) > 2^{i-1}$ ($i = 1, 2, \ldots, n$; 2^n is the ID space size). A successor is one of the closest nodes to u. When v is a successor, then $\rho(u, d) - \rho(u, v) \ge 1$ (case of local routing, $\delta \ge 1$ in Theorem 1). When the next-hop node v is a finger, then $\rho(u, v) \ge \rho(u, d)/2$ (case of global routing, $k = 2$ in Theorem 2).

According to Theorems 1 and 2, CR may follow the same rules of progressive routing as the underlying DHT. To satisfy these rules, only the next-hop nodes are used. It is enough to provide the same dependability and not to degrade the performance upper bounds. We discuss more efficient routing in Sect. III-D.

C. Constructing Cycles

Inequations (1) and (2) do not provide a direct way for constructing appropriate cycles. The dependence on d means that there should be a cycle at a node u for any destination d. In this section, we consider the problem of constructing a cycle appropriate for many destinations.

DHT Lookups: A consequence of bidirectional P2P communications. The method caches paths discovered in lookups.

Let a lookup packet follow the path $s \rightarrow^+ d$ collecting some visited node IDs ($v_1, \ldots, v_{i-1}$). They are included into the acknowledgment that d sends to s. Similarly, some node IDs ($v_i, \ldots, v_{l-1}$) are collected in the backward trip $d \rightarrow^+ s$. According to the theorems above, the cycle

$$s \rightarrow v_1 \rightarrow^+ \ldots \rightarrow^+ v_{i-1} \rightarrow^+ v_i \rightarrow^+ \ldots \rightarrow^+ v_{l-1} \rightarrow^+ s$$

can be used later for dependable routing to any d' such that $\rho(v_1, d') < \rho(s, d')$.

Look-Ahead: Let u maintain $\mathcal{C}_u$ such that for any $c \in \mathcal{C}_u$

$$c = \big(u \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{l-1} \rightarrow^+ u\big). \tag{3}$$

Nodes u, v_1, $\ldots v_{l-1}$ are consecutive neighbors ($v_1 \in T_u$, $v_2 \in T_{v_1}$, $\ldots$) and $l - 2$ is the level of look-ahead (different for different nodes). NoN-greedy routing [4], [8] is a particular case of (3) when $l = 3$ and u maintains all cycles $u \rightarrow v_1 \rightarrow v_2 \rightarrow^+ u$ (for all neighbor's neighbors).

To construct a cycle in form (3) u asks its neighbor v. It replies with some neighbors w. Then u constructs cycles $u \rightarrow v \rightarrow w \rightarrow^+ u$. Also v can reply with cycles $v \rightarrow w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_{l-1} \rightarrow^+ v$. In this case, u constructs cycles $u \rightarrow v \rightarrow w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_{l-1} \rightarrow^+ u$.

The look-ahead construction can be easily combined with DHT lookups; a lookup packet collects only the first n nodes of the route $s \rightarrow^+ d \rightarrow^+ s$.

These methods have the following properties.

Using existing overlay paths: This property preserves the routing dependability (in terms of Theorems 1 and 2). Moreover, it provides a kind of synchronization among nodes, since the nodes of the route are likely to use the same path that has been already in use.

Incrementality: CR can start from an overlay without any knowledge of cycles. Either passive or proactive cycle construction is possible. The former piggybacks on existing lookup packets running in the overlay. The latter introduces extra packets for cycle construction.

No substitution of local routing tables: A direct IP contact is more efficient than a multi-hop path. If v provides its IP to u, then u puts (v, IP_v) into T_u. However, when node IPs are not accessible the routing table cannot be updated, and the cyclic structure is a way to keep the additional information.

Overhead control: Cycle nodes are collected in lookup packets; it causes some overhead in terms of packet size. As we discuss in Sect. III-E, CR needs only short cycles bounded with $O(\log N)$ hops. Moreover, it is not required to keep all intermediate nodes; some of them may be skipped. For instance, in the look-ahead method only the first $l-1$ nodes are collected and a node can vary l.

D. Finding the Most Efficient Cycle

According to Theorems 1 and 2, the selection of an appropriate cycle depends only on the distance between the current node, next-hop node and destination. Algorithm 1, however, supports more efficient routing. It relates to the definition of "d' is close to d" and depends on routing criteria, e.g., within few hops, with low latency or with high success probability.

The simplest case is exact matching $d' = d$ when an efficient cycle contains the destination. Another way defines a neighborhood with $H > 0$; in an efficient cycle, d' is a nearby node, i.e., $\rho(d', d) \le H$.

Let u find such a cycle that

$$\begin{aligned} &c = (u \to v_1 \to^+ \cdots \to^+ v_{l-1} \to u) \text{ and} \\ &\rho(d', d) \text{ is minimum among all } c \in \mathcal{C}_u,\ d' = v_i \in c. \end{aligned} \tag{4}$$

The same is in NoN-greedy routing [4] for $l = 3$ in (3), improving the efficiency from $O(\log N)$ to $O(\log N / \log\log N)$ hops in skip-graphs and small-world graphs.

Then u has to decide either using c or the underlying DHT. Basically, c is preferable when $\rho(d', d) \ll \rho(v, d)$. Some other parameters can also be important.

Assume that the underlying DHT provides geometrically progressive routing. If u forwards to v, the number of hops is estimated as $L_{\mathrm{DHT}} = \log_k n_{ud} + 1$ (similarly to the proof of Theorem 2). If u uses c then routing takes either $L_{\mathrm{CR}} = i$ hops when $\rho(d', d) \le H$ (i.e., $i-1$ hops to reach $d' = v_i$ and then one hop to d), or $L_{\mathrm{CR}} = i + \log_k n_{d'd}$ hops when $\rho(d', d) > H$. Accordingly, c is preferable if $L_{\mathrm{CR}} < L_{\mathrm{DHT}}$.

Some DHTs estimate the round trip time, $\mathrm{RTT}(u, v)$ and $\mathrm{RTT}(u, v_1)$. Taking the latency into account, c is preferable if $L_{\mathrm{CR}} \cdot \mathrm{RTT}(u, v_1) < L_{\mathrm{DHT}} \cdot \mathrm{RTT}(u, v)$. It is similar to the proximity route selection [9].

Some P2P systems rank nodes according to their behavior, $0 \le r_v(u) \le 1$. For instance, $r_v(u)$ is the probability that v successfully serves lookups sent from u. Then a cycle rank is $r_c = \prod_{j=1}^{i} r_{v_j}$, and c is preferable if $r_c > r_v r_{\min}^{L_{\mathrm{DHT}}-2}$, where $r_{\min}$ is the default rank for unknown nodes.

Applying the criteria above, we can modify (4) in different ways. For instance, minimize i or $i \cdot \mathrm{RTT}(u, v_1)$ (or maximize $\prod_{j=1}^{i} r_{v_j}(u)$) subject to $\rho(v_i, d) \le H$.

E. Maintaining Cycles

Each node u maintains the cyclic structure $\mathcal{C}_u$ as a cache, inserting new cycles and removing invalid cycles.

New cycles: Constructing methods are described in Sect. III-C. According to Sect. III-D a node inserts only efficient cycles. The same efficiency criteria can be applied except that there is no destination d fixed. An efficient cycle is short ($O(\log N)$ hops) and its first node is IP-close ($\mathrm{RTT}(u, v_1)$ is small, as in the proximity neighbor selection [9]). Path-specific metrics can also be used, e.g., the round trip time along the cycle.

Memory cost: Given the typical bound $|T_u| = O(\log N)$. Preferably, each neighbor should be first in several cycles since finding a good cycle needs a comprehensive set of candidates. We assume that the number of such cycles is $O(\log N)$ because of the same reason as in geometrically progressive routing. An efficient cycle is short and consists of $O(\log N)$ nodes. As a result, a reasonable bound for IDs a node maintains is $O(\log^3 N)$.

Note, however, that a node may vary the cyclic structure size according to available resources and independently on other nodes. CR works even if there are nodes that do not maintain any cyclic structure.

Cycle rank: A node ranks cycles according to the routing history. Low-ranked cycles are removed or replaced. It takes into account the cycle usage rate, cycle success probability, cycle round trip time, and cycle stability. Note that these metrics can be estimated passively using regular lookups.

Invalid cycles: In a DHT, nodes ping neighbors for availability and stability. Similarly, cycle checking uses lookups, either passively or proactively. As was shown above, a node maintains $O(\log^2 N)$ cycles compared to $O(\log N)$ neighbors. Nevertheless, when a node checks a cycle then the neighbor is tested too. While pinging a node does not provide a new neighbor, checking a cycle can produce a new one.

In dynamic environments, a cycle is less stable than a node. Let p be the probability that a node is alive. A cycle of $n > 1$ nodes is alive with the lower probability p^n. However, cyclic structures gradually accumulate stable cycles since most of the nodes in real P2P systems are long-lived [14].

IV. Simulation Results

CR-Chord was implemented as an extension of the MIT Chord simulator and simulation experiments were performed to compare Chord vs. CR-Chord in the presence of malicious DHT nodes and under churn. The results showed that CR-Chord has better lookup availability.

TABLE I
LOOKUP SUCCESS IMPROVEMENT AND LOOKUP FAILURE REDUCTION ($N = 10^3$, $L = 0.01N^2 = 10^4$).

Malicious nodes $M = fN$, %	10%		20%		30%		40%		50%		Average	
R, s^{-1}	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2
S_{chord}	6724.9	5141.0	4353.8	3110.1	2673.9	1900.4	1501.1	994.7	808.2	534.9	3212.4	2336.2
$\Delta_{\text{c}}(S_{\text{chord}})$, %	23.6		28.6		28.9		33.7		33.8		29.7	
S_{cr}	7958.4	7557.9	5985.4	5262.2	4197.9	3526.2	2617.3	2042.3	1518.5	1154.9	4455.5	3908.7
$\Delta_{\text{c}}(S_{\text{cr}})$, %	5.0		12.1		16.0		22.0		23.9		15.8	
$\frac{S_{\text{cr}} - S_{\text{chord}}}{S_{\text{chord}}}$, %	18.3	47.0	37.5	69.2	57.0	85.6	74.4	105.3	87.9	115.9	55.0	84.6
F_{chord}	3275.1	4859.0	5646.2	6889.9	7326.1	8099.6	8498.9	9005.3	9191.8	9465.1	6787.6	7663.8
$\Delta_{\text{c}}(F_{\text{chord}})$, %	48.4		22.0		10.6		6.0		3.0		18.0	
F_{cr}	2041.6	2442.1	4014.6	4737.8	5802.1	6473.8	7382.7	7957.7	8481.5	8845.1	5544.5	6091.3
$\Delta_{\text{c}}(F_{\text{cr}})$, %	19.6		18.0		11.6		7.8		4.3		12.3	
$\frac{F_{\text{chord}} - F_{\text{cr}}}{F_{\text{chord}}}$, %	37.7	49.7	28.9	37.0	20.8	20.1	13.1	11.6	7.7	6.6	21.6	23.8

In this section, we use a part of the data produced with the Chord and CR-Chord simulation in [15]. Compared to it, we provide further analysis of the basic CR properties. Note that [15] does not take into account the IP-level reachability restrictions; and produces more pessimistic estimates for the CR method.

A. Chord and CR-Chord implementations

Both Chord and CR-Chord use flexible routing table maintenance when a node keeps n mandatory fingers and also n_{af} additional fingers to know more about the network.

CR-Chord implements cyclic routing when an appropriate cycle (if any) is piggybacked into the packet. Only short cycles are stored for reuse; the number of hops is at most $2\log_2 N$, where a node approximates $\log_2 N$ as $\min(f, n)$, f is the number of fingers and n is the number of ID bits.

CR-Chord supports a kind of multipath routing when m_{d} duplicate packets are sent using fingers closest to the destination. This mechanism is used for constructing cycles (see the DHT lookups method in Sect. III-C).

B. Simulation setup

Chord DHT of size $N = 1000$ nodes is simulated. The Chord ID space is of $n = 24$ (2^{24} node IDs). Other parameters are $n_{\text{af}} = 12$ and $m_{\text{d}} = 3$. Each simulation experiment is executed 10 times for averaging.

We assume that malicious nodes ignore all data lookups for which they are responsible. That is, data at such nodes are lost. They process correctly all other requests trying to hide the malicious activity.

Network construction consists of three steps. First, a Chord DHT of G good nodes is constructed; nodes are joining randomly. Second, $D = 100N = 10^5$ documents are distributed uniformly. Third, $M = fN$ malicious nodes join the network at the same time, where $G + M = N$ and f is the fraction of malicious nodes (varied from 5% up to 50%).

There are 100 documents per a node on the average but about $100M$ documents are lost. Good nodes initiate randomly $L = 0.01N^2 = 10^4$ requests in total with rate 1.0 s^{-1} (requests per second). The standard Chord stabilization procedure is executed every 30 s.

Churn consists of node joins and leaves. To preserve the ratio of good and malicious nodes, any node join is accompanied with a node leave and vice versa; those happen with rates $R = 0, 0.02, 0.2$ s^{-1}. When the lookup rate is 1.0 s^{-1} as above, a churn join/leave pair happens every 5 lookups for $R = 0.2$ s^{-1} and every 50 lookups for $R = 0.2$ s^{-1}.

C. CR-Chord vs. Chord

Let $S_{\text{chord}}(f, R)$ and $F_{\text{chord}}(f, R) = L - S_{\text{chord}}(f, R)$ be counters of successful and failed lookups for Chord. Similarly, $S_{\text{cr}}(f, R)$ and $F_{\text{cr}}(f, R) = L - S_{\text{cr}}(f, R)$ are for CR-Chord.

Table I presents the estimates of S and F when varying the fraction of malicious nodes. The relative lookup success improvement and relative lookup failure reduction,

$$\Delta(S) = \frac{S_{\text{cr}} - S_{\text{chord}}}{S_{\text{chord}}} \quad \text{and} \quad \Delta(F) = \frac{F_{\text{chord}} - F_{\text{cr}}}{F_{\text{chord}}},$$

are also given. In these relative estimates, Chord is a base.

Two cases—with churn ($R = 0.2$) and without churn ($R = 0$)—are considered. The effect of churn is estimated with the relative variation

$$\Delta_{\text{c}}(x) = \frac{x(f, 0) - x(f, 0.2)}{x(f, 0)} \quad \text{for } x = S, F,$$

where $R = 0$ is a basement.

Both in Chord and CR-Chord, the lookup success (failure) rate decreases (increases) with more malicious nodes and churn. CR-Chord always has better values for these rates. Interestingly that CR-Chord under churn ($R = 0.2$) outperforms Chord in stable networks ($R = 0$). Fig. 2 is from [15] and graphically shows the estimates of lookup failure rate.

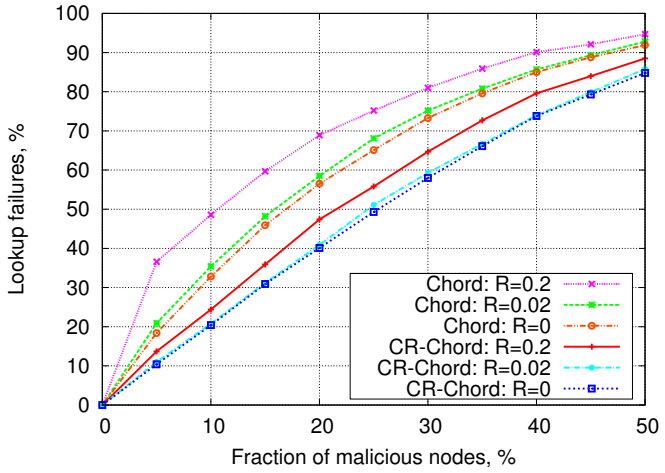

Fig. 2. Lookup failure rates for different churn rates. CR-Chord always outperforms Chord. The CR efficiency is lower for larger f and R.

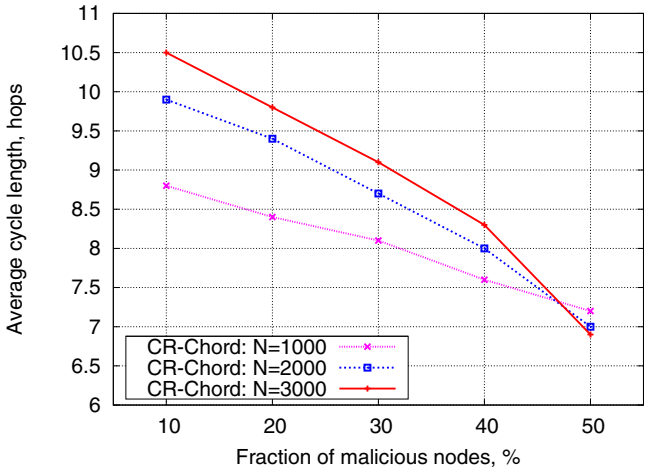

Fig. 4. The average cycle length (in hops) for different network sizes ($R = 0$, $L = 0.01N$). In a larger network, cycles are longer. Cycles become shorter in malicious networks.

When the fraction of malicious nodes is small (5–10%) the CR-Chord lookup failure rate is almost twice lower. When the fraction of malicious nodes is large (40–50%) the CR-Chord lookup success rate is about twice higher.

Churn affects the CR-Chord lookup availability less. In particular, $\Delta_c(S_{\text{chord}}) > \Delta_c(S_{\text{cr}})$ for all fractions of malicious nodes, and $\Delta_c(S_{\text{chord}})/\Delta_c(S_{\text{cr}}) \approx 2$ on the average. In terms of the lookup failure rate, $\Delta_c(F_{\text{chord}})$ becomes slightly lower than $\Delta_c(F_{\text{cr}})$ only for large f. On average, $\Delta_c(F_{\text{chord}})/\Delta_c(F_{\text{cr}}) \approx 1.5$.

In fact, CR accelerates the routing mechanism of underlying DHT. The effect is achieved with a small set of cycles per node (Fig. 3). The length of cycles is short (Fig. 4).

The CR efficiency is reduced for large f and R. The reason is that underlying DHT lookups do not allow constructing good cycles (although good paths do exist). As a result, there is a lack of cycles constructed at a node (see Fig. 3), they are too short and hence less global (see Fig. 4), and the difference between Chord and CR-Chord becomes small.

As was shown in [15], known methods like flexible routing table maintenance (parameter n_{af}) multipath routing (parameter m_{d}) become inefficient for finding good paths when f or R are large. From this point of view, the look-ahead method seems more promising for constructing cycles.

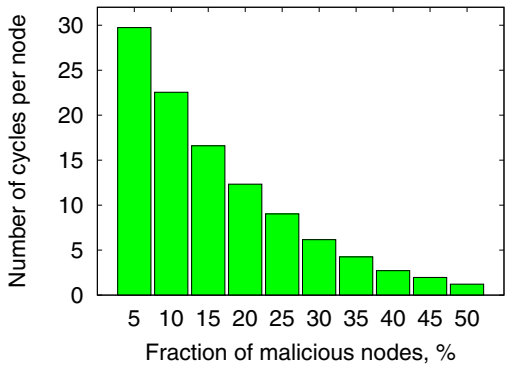

Fig. 3. The average number of cycles per node ($N = 10^3$, $R = 0$). There is a lack of cycles constructed in very malicious networks.

V. Discussion on Advanced Capabilities

The basic CR design is an additional routing mechanism to deal primarily with the IP-level reachability restrictions. It can be combined with known methods such as flexible routing table maintenance and multipath routing. The simulation showed that this combination also works efficiently in networks with malicious nodes and churn. This section briefly discusses some advanced capabilities that CR can provide.

A. Global and Local Routing

In many DHTs, routing to a destination neighborhood is an important point. Let us mention two reasons.

First, DHT routing is usually key-based, not nodeID-based, performing routing to a node responsible for a given key. Since a node is responsible for all keys close to its ID, let a message arrive to a node close to the key. Moreover, DHT replication techniques distribute data among nearby nodes.

Second, a node typically knows its neighborhood well since many nearby nodes are in the routing table. As an example, the Chord DHT keeps several successors and predecessors. Consequently, one hop is needed when a message is at a nearby node.

Therefore, we distinguish two parts of P2P routing, *global* and *local*. In global routing, a message is delivered close to the destination. In local routing, the destination is at a nearby node. By design, CR aims at global routing (Fig. 5). When an appropriate cycle is found, then global routing happens, crossing a large part of the overlay and arriving to a nearby node d'.

Cyclic routing can also act as local when gaps appear in routing tables because of IP-level reachability restrictions, e.g.,

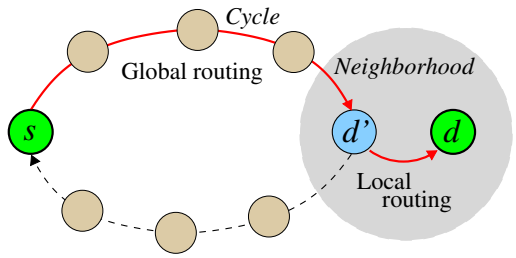

Fig. 5. In a lookup path $s \rightarrow^+ d$, CR acts primarily as global routing $s \rightarrow^+ d'$; local routing $d' \rightarrow^+ d$ takes typically one hop.

a nearby node is located behind a NAT. Moreover, cycles provide some information to prioritize routing. For instance, a neighbor is good if cycle success rate is high for those cycles that this neighbor starts.

B. Cyclic vs. acyclic paths

In principle, a node may maintain paths to a set of destinations instead of cycles, as in link-state routing protocols [16] or MPLS networks [17]. In link-state routing, a node keeps a graph of the whole network. Global graph analysis calculates entries (d, IP_v), where v is the-next hop node of the best path to d. In MPLS networks, communication between s and d is preceded with construction of a path $s \rightarrow^+ d$, where all nodes know to which node to forward a message. These approaches are essentially global and do not scale well.

On the other hand, basic DHT routing uses a local view of the network when a node knows a limited set of neighbors. This approach scales well but in some cases more knowledge about the network is needed.

The CR design allows a trade-off between the local and global cases. When a node does not use cyclic structure, only the local view is presented. With cyclic structure a node knows more, up to the extreme case when all cycles are known.

A cycle $s \rightarrow^+ d \rightarrow^+ s$ contains more information than a non-cyclic path since both forward and backward paths are presented. First, it introduces explicit support for bidirectional communication. Second, there is no fixed destination in a cycle compared with a source-destination path, and the same cycle can be used for routing to many destinations.

C. Multi-path routing

Multi-path routing is a way to deal with problems such as path congestion, node failures, high path latency, and the presence of malicious nodes [5], [12], [18].

In DHTs, when s experiences a problem with the best next-hop, another one is used. This mechanism is enhanced in CR. In fact, CR provides a way to route messages around overloaded, failed or malicious nodes.

First, a node can duplicate a message using both the most efficient cycle and DHT next-hop node (see Sect. III-D). Basically, only a source node makes this duplication (for example, in CR-Chord [15]). However, other strategies are possible.

Second, there can be several appropriate cycles in $\mathcal{C}_s$ for a given destination d. They provide a precomputed pool of alternate paths $s \rightarrow^+ d$. That is, s sends duplicates through several cycles like in the neighbor-set anycast [12] or the constrained multicast [5]). Moreover, s can sort alternate cycles according to their efficiency, as in the sorted backup paths technique [5].

The CR method also supports disjoint paths [18]. A node s treats alternate cycles to d as semi-disjoint if their subpaths $s \rightarrow^+ d$ do not share common intermediate nodes. The prefix "semi" appears since s does not necessarily know all nodes of these cycles. To date, although disjoint paths are very valuable for multi-path and secure routing, most of P2P systems do not support them.

In multicast communication, a node s sends a message to a set D of destinations. In the CR method, s forms a pool $\mathcal{M}_s(D) \subseteq \mathcal{C}_s$ of appropriate cycles that covers all destinations. The message is sent using these cycles. Although in the worst case the size of $\mathcal{M}_s(D)$ can be equal to $|D|$, several destinations can be reached with one cycle. Hence, the duplication is reduced.

D. Security

The CR efficiency for networks with malicious nodes is shown in Sect. IV and in our simulations [15].

Recall that the CR method reduces the use of IP addresses. Even if a node restricts access to its IP address, other nodes still can gain knowing that the node appears as an intermediary in some lookups. Consequently, a DHT equipped with CR is less exposed to IP-level attacks, such as Denial-of-Service (DoS).

The CR-method supports bidirectional communications. The security level is higher when both directions are independent, and traffic goes through disjoint paths. Similarly to semi-disjoint cycles, a source node selects a cycle with independent forward and backward directions.

In CR method, a node has more possibilities to detect incorrect or suspicious routes. For instance, let a node u appear at s in a large fraction of cycles. This fact signals that u becomes dangerous for s. At least, u is a bottleneck; in a worse case, u can disrupt communications of s with other nodes. Therefore, the cyclic structure helps s to detect such a risky situation in advance.

Assume a node s detects that a node u is malicious. In a basic DHT, s can only remove u from its routing table. In the CR method, s also removes all cycles containing u as well as s can decrease the priority of those neighbors that routed messages to u (they are first-hop nodes in the cycles with u).

VI. Conclusion

Recent progress in DHT routing made possible many practical P2P-based applications in the Internet. However, they work poorly when no direct IP connectivity is available to some nodes (e.g., located behind a NAT or firewall) and in the presence of malicious or overloaded nodes.

This paper introduced the design of Cyclic Routing (CR), a method that improves routing over basic DHT algorithms.

Experimental analysis is provided using data from simulations [15] for the Chord DHT enhanced with CR.

In simulations, malicious nodes dropped lookup packets but there were no restrictions on IP addressing. The results suggest that CR improves the Chord lookup availability. For instance, when the number of malicious nodes is small (5–10%) CR-Chord has almost twice lower lookup failure rate. When the number of malicious nodes is large (40–50%) CR-Chord has about twice higher lookup success rate.

The CR method does not substitute known methods of flexible routing table maintenance and multipath routing but complements them when the IP-level reachability fails. The CR method allows a node to have a broader view on the global network. Since CR only uses node IDs, it is applicable even when a routing table entry insertion is not possible, e.g., when the node's IP address is not globally routable or frequently changes.

Cyclic routing generalizes the idea of look-ahead approach to DHT routing, maintaining a trade-off between the look-ahead level and available resources. Therefore, CR takes a place between two extreme cases: 1) DHT routing using only neighbors (local) and 2) shortest-path routing (global).

Using additional knowledge of the global network, cycles enable multiple-path packet delivery and routing around malicious or overloaded nodes. Instead of selecting a path based on neighbors only, CR exploits paths already discovered in the overlay. From this point of view, cyclic structures at nodes accumulate stable efficient paths in the overlay improving its routing.

REFERENCES

[1] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica, "Non-transitive connectivity and dhts," in *Proc. of the 2nd USENIX Workshop on on Real, Large Distributed Systems (WORLDS'05)*, San Francisco, CA, Dec. 2005, pp. 55–60.

[2] M. Castro, P. Drushel, Y. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer networks," Microsoft Research, Technical Report MSR-TR-2002-82, 2002.

[3] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 99–114.

[4] G. S. Manku, M. Naor, and U. Wieder, "Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks," in *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2004, pp. 54–63.

[5] B. Y. Zhao, L. Huang, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz, "Exploiting routing redundancy via structured peer-to-peer overlays," in *Proc. of The 11th IEEE International Conference on Network Protocols (ICNP '03)*, Atlanta, GA, Oct. 2003, pp. 246–257. [Online]. Available: citeseer.ist.psu.edu/article/zhao03exploiting.html

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

[7] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003. [Online]. Available: citeseer.ist.psu.edu/kaashoek03koorde.html

[8] M. Naor and U. Wieder, "Know thy neighbor's neighbor: Better routing for skip-graphs and small worlds," in *The Third International Workshop on Peer-to-Peer Systems (IPTPS).*, 2004. [Online]. Available: citeseer.ist.psu.edu/article/naor04know.html

[9] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. of ACM SIGCOMM'03*. New York, NY, USA: ACM Press, Aug. 2003, pp. 381–394.

[10] H. Zhang, A. Goel, and R. Govindan, "Incrementally improving lookup latency in distributed hash table systems," in *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2003, pp. 114–125.

[11] B. Leong, B. Liskov, and E. Demaine, "Epichord: parallelizing the chord lookup algorithm with reactive routing state management," in *Proceedings of the 12th International Conference on Networks 2004 (ICON 2004), Singapore*, Nov. 2004, pp. 270–276.

[12] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proc. of the 5th USENIX Symposium on Operating System Design and Implementation (OSDI 2002)*. Boston, MA, USA: ACM Press, Dec. 2002, pp. 299–314.

[13] M. Castro, M. Costa, and A. Rowstron, "Performance and dependability of structured peer-to-peer overlays," Microsoft Research, Technical Report MSR-TR-2003-94, Dec. 2003.

[14] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 189–202.

[15] D. Korzun, B. Nechaev, and A. Gurtov, "CR-Chord: Improving lookup availability in the presence of malicious DHT nodes," HIIT Technical Report 2008-2, Dec. 2008. [Online]. Available: http://www.hiit.fi/nrg-publications

[16] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[17] V. Sharma and F. Hellstrand, "Framework for multi-protocol label switching (MPLS)-based recovery," IETF, RFC 3469, Feb. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3469.txt

[18] M. Srivatsa and L. Liu, "Vulnerabilities and security threats in structured overlay networks: A quantitative analysis," in *ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 252–261.