Computer Networks and Communications
https://ojs.wiserpub.com/index.php/CNC

Article

Analysis of Path Convergence in Chord DHT

Vladimir Rocha*, Daniel Czeresnia and Carlo Kleber da Silva Rodrigues

Center for Mathematics, Computing and Cognition, Federal University of ABC, Santo André, SP, Brazil
E-mail: vladimir.rocha@ufabc.edu.br

Received: 23 April 2024; **Revised:** 12 July 2024; **Accepted:** 15 July 2024

Abstract: Chord is a Distributed Hash Table widely used for its efficiency in searching for information. The efficiency of this structure relies on creating short paths of O($\log_2$ n) between two nodes, in which n is the number of nodes. To enhance efficiency, several studies use replication in the nodes belonging to the network, assuming that searches will converge in these replicated nodes. This work proposes a convergence formula and analyzes the number of searches converging on nodes for different network sizes (small, medium, and large), up to one million nodes. The experiments show that the convergence creates three zones and the results support the replication techniques from previous studies and demonstrate that it is feasible to replicate in nodes that were not considered in these studies.

***Keywords*:** Chord, path convergence, Distributed Hash Table, replication

1. Introduction

The Distributed Hash Table is a widely known distributed structure used in various scenarios as a basis for efficiently searching for information in distributed systems, for example, multimedia systems [1], cloud systems [2], multiagent systems [3], blockchain systems [4, 5], storage systems [6], IoT systems [7], among others [8]. In the context of computer networks, the Distributed Hash Table is relevant and used for creating efficient routing and forwarding tables [9], for establishing low-latency connections [10], for optimizing the network topology for multicast [11], among other situations.

Chord is an implementation of the Distributed Hash Table with a ring-shaped overlay network, where each node in the network has pointers to its successor in the ring and to other nodes. To find information, it is necessary for the search request to traverse a path of nodes, starting at any node, called the origin, and ending at the node responsible for the information requested, called the destination. In other words, the request will be forwarded through several nodes until it finds the destination node [12].

To improve search efficiency, different works promote the replication of the information requested in some nodes of the network, whether in successor nodes or in nodes that are part of the path traveled by the request, assuming that searches converge on them. In this sense, by replicating the information in the proximity to the destination, the path traveled is reduced, allowing it to be found faster [13].

Although replication in the proximity to the destination is used to improve search efficiency, how close to the destination should the information be replicated to improve its recovery? Moreover, is it still possible to improve search efficiency if the replication is not that close? Besides, does the size of the network influence the recovery efficiency? The motivation for answering these questions is to understand where to replicate and if in these locations it is possible to improve search efficiency, which is not covered by previous researches that use the proximity word, but without defining and measuring it. To answer the questions mentioned above, in this work, we examine the paths traversed by the searches (creating a tree of paths), develop a formula for convergence for different proximities, and calculate its value for different network sizes.

The contributions of our work are:

(i) To validate the replication of previous research works;

(ii) To develop a convergence formula for the search paths;

(iii) To know the exact convergence of the nodes, enabling a more insightful replication;

(iv) To show that it is possible to replicate in nodes that were not considered in previous researches, supporting the replication of the information requested;

(v) To test the convergence in large networks.

The article is structured as follows: Section 2 details the Chord Distributed Hash Table, illustrating the process of inserting and searching for information. Section 3 examines previous research on convergence and replication. Section 4 introduces the tree of paths and the convergence formula utilized in this study. Section 5 outlines the experiments, results, and analyzes convergence across various network sizes. Lastly, Section 6 discusses conclusions and future research directions.

2. The Chord Distributed Hash Table

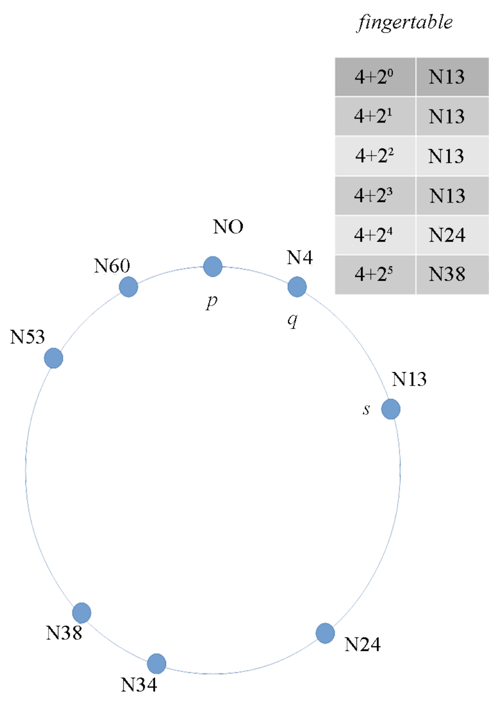
A Distributed Hash Table (DHT) is a structure that allows it to perform the functions of a normal hash table, but in a distributed manner. Thus, like the normal hash table, the DHT basically has two functions: *get* and *put*. In the first one, the key is searched to obtain the value associated with it (that was stored in the structure). In the second one, a key-value pair is stored in the structure. However, in the DHT, these functionalities are carried out in a distributed and efficient manner among the nodes belonging to the network [14].

Chord is a DHT implementation composed of n logically connected nodes forming a ring-shaped overlay network and ordered in ascending order by identifiers. In Chord, a node q has: a unique identifier q.ID; pointers to its predecessor p and successor s; and is responsible for storing key-value pairs, whose key transformed into an identifier corresponds to the values that are between p.ID+1 and q.ID [15]. Figure 1 shows an example of a Chord network with 8 nodes, in which the node q has ID = 4, with the predecessor p with ID = 0 and the successor s with ID = 13. In addition, it is responsible for storing identifiers from 1 to 4.

Besides the predecessor and successor, each node has pointers to other nodes in the ring. These pointers are stored in a vector called fingertable. Each position i of the node vector with identifier ID is calculated by the formula fingertable[i] = ID + 2^i, which points to the node whose identifier succeeds the calculated value [16]. In Figure 1 the fingertable of the node q is shown with their respective pointers.

The process to search for a key in Chord (i.e., the *get* function) is as follows. Some node in the ring must receive the search request—any node can receive the request. When the node q receives a request *req*, it will transform the key into an identifier k (using a hash function). For example, a string key 'Ubuntu.iso' is transformed into an identifier k using the MD5 hash function, i.e., k = MD5('Ubuntu.iso'). Then, the node q will search in its fingertable for the node t nearest to k, forwarding the request req to node t. This forwarding will go through other nodes, forming a path, until it reaches the one responsible for storing the identifier k. In this context, the search will only need O($\log_2$ n) forwards to find the node responsible for the key. Note that in each request forwarding, there is an exponential advance in the search space, very similar to binary search in an array [17]. To understand the logarithmic amount of forwards, consider a fingertable with size m, such that a node ID $\in [0, 2^m)$, and two nodes: s, the source, and d, the destination. By the fingertable formula, in s' fingertable must be a node c such that c.ID $-$ d.ID $\leq 2^{m-1}$ in which the 1 in the exponent denotes the first forwarding. After log n forwardings, we get $2^{m - \log n} = 2^m/n$, which represents the distance from the last forwarding node to the destination node d. At this distance, since IDs are uniformly assigned in the ring, there could be one node, in expectation, giving, in total, $\log_2 n + 1$ or $O(\log_2 n)$ forwardings to reach the destination node [12]. Figure 2 shows the search path for ID = 60 from the source node q, where each arrow corresponds to a search request that was forwarded.

The process to insert a pair key-value in Chord (i.e., the *put* function) is as follows. A node q must receive the insertion request *req* and transform the key into an identifier k (using a hash function) in the same way as in the search. Then, node q uses the search process mentioned above to find the destination node d responsible for k and executes the insertion of k (with its corresponding value) in d. Note that, as the insertion uses the search process, it will need $O(\log_2 n)$ forwardings as well.



$4+2^0$	N13
$4+2^1$	N13
$4+2^2$	N13
$4+2^3$	N13
$4+2^4$	N24
$4+2^5$	N38

Figure 1. Chord with 8 nodes

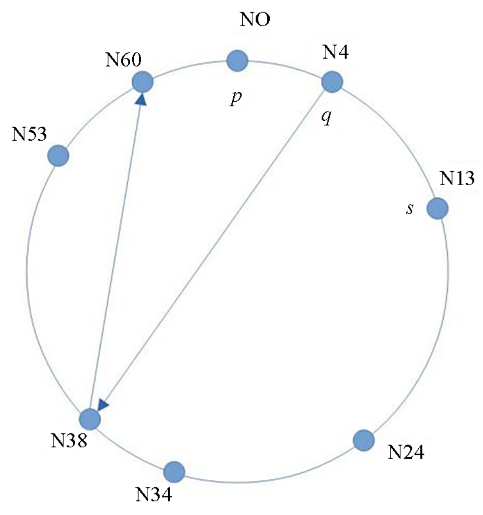


Figure 2. Searching for ID = 60 from node q

3. Related work

In this section, we present the related works, which have been divided into two parts. In the first part, we analyze the works that assume the existence of path convergence and focus on information replication to improve search efficiency. In the second part, we analyze the works that use convergence to shorten the path, consequently improving the efficiency of the search for information.

3.1 *Information replicas*

In [18], the authors replicate the information in some nodes of the fingertable responsible for the key requested. The intention is to distribute the information among the network nodes evenly, given that these nodes are at different distances from the destination node (by the fingertable' exponential formula), and therefore in different places on the ring. The experiments were carried out on small networks of 512 to 4096 nodes, in dynamic scenarios where the nodes could die, and replicating the information in 12 nodes. As a result, a maximum decrease from 37 to 25 request forwards was obtained when the node failures were very high.

In [19], the authors use replication in a highly dynamic vehicular network scenario. In this scenario, each vehicle corresponds to a DHT node, which can enter or leave the network at any time, generating continuous changes in the paths used to find the information requested. The information replicated is carried out similarly to the previous work, but (a) it is replicated in all the nodes of the fingertable of the responsible node (not just in some of them); and (b) it considers the physical distance between the nodes (nodes closer geographically create links among them). The experiments were carried out on small networks of 100 to 500 nodes, reducing the forwarding of the search for information from 30 to 17.

In [20], the authors extend the fingertable by adding the nodes that have the replica of the searched information. The replication is carried out in two manners. In the first one, as soon as the information is stored, it is replicated in the predecessors and successors of the responsible node. In the second one, the information can be replicated in the node that made the search request assuming that other searches will converge on them. The experiments were carried out on small networks of 50 to 500 nodes, with a replication probability of 70%, reducing the forwarding of the search for information from 6.25 to 2.5.

3.2 *Path convergence*

In [21], the authors mention that the paths traveled by searches for the same key, from different origin nodes, will intersect in the proximity of the node responsible for the key. According to the authors, this occurs because, in the proximity of the destination, few nodes are part of the paths leading to the node responsible for the key. With this, it is possible to replicate the information near the destination to decrease the request forwarding and find the information faster. The experiments were carried out on a small network of 1000 nodes, reducing the forwarding of the search for information from 5.7 to 3.2 using replication.

In [22], the authors explore convergence by dividing the network into two disjoint sets of nodes (called domains) and measuring the exit points. The exit points are those nodes from one domain whose next request forwarding arrives at a node from the other domain. In the work, it is assumed that the fewer exit points, the greater the convergence. In the experiments carried out, a medium network was created with 65,536 nodes and with domains of different sizes (small, medium, and large). The results showed that convergence is greater in small and large domain sizes, and smaller in medium domain sizes. In addition, it was also found that by choosing the closest node in terms of latency (in the path), the time to find the information is reduced.

Finally, in [23], the authors consider the convergence of two or more search paths at a common node, close to the nodes that originated the searches. In the work, two strategies were analyzed, both aiming to reduce the path to find the information. The first strategy was to replicate in nodes with high convergence, and the second strategy was to replicate in any node that receives the forwarding. In the experiments carried out, a medium network was created with 16,000 nodes and measured how many replicas are necessary for the strategies to behave in the same way. From the results, it was obtained that the second strategy needs five times more replicas to reach the same amount of forwarding as the first strategy and that, and even then, the first strategy has slightly smaller paths than the second.

It is important to mention that, both for the cited works that focus on replication and for those that use path convergence, the main differential of the work proposed here is that convergence is analyzed for all nodes of the paths (not only in some of them), generating a tree of paths, which will allow for replication in nodes that were not considered in the cited works. Also, unlike previous works that only mention a general convergence statement, with the tree, it is possible to know the exact convergence at each level, allowing to perform a more informative replication. Additionally, the convergence was analyzed in large networks, as well as small and medium ones. Considering the studies above, Table 1 compiles the replication key aspects and the differences from our work.

Table 1. Synthesis of literature works and main differences

Reference	Brief Description	Difference with Previous Research
[18]	Replicate information at different positions of the ring, using the destination fingertable. Tested in small networks of 4096 nodes.	• Develop a convergence formula and measure the convergence for different proximity values, spanning all nodes in the search path.
[19]	Replicate information on all nodes stored in the destination's fingertable. Tested in small networks of 500 nodes.	• Test the convergence with different small (up to 5000 nodes) and medium (up to 100,000 nodes) size networks.
[20]	Replicate information in the proximity (predecessor and successor) of the destination node. Tested in small networks of 500 nodes.	• Test the convergence in large networks (up to 1,000,000 nodes).
[21]	Replicate information in the proximity of the destination node. Tested in small networks of 1000 nodes.	• Analyze the replication for different proximity values.
[22]	Replicate information in exit points of a group closer to the destination node. Tested in medium networks of 65,536 nodes.	• Test the convergence in small and large networks (up to 1,000,000 nodes).
[23]	Replicate information close to the origin node. Tested in medium networks of 16,000 nodes.	• Test the convergence in medium and large networks (up to 1,000,000 nodes).

4. Path convergence in Chord

As mentioned in the Introduction, to improve search efficiency, the nodes where the information will be replicated should be those that allow reducing the path traveled by the search (through requests forwarding) until finding the information. A path consist of the origin node, all nodes that forward the search request, and the destination node.

As shown in related works, one alternative is to use replication in the nodes belonging to the path traveled by the requests. In this sense, considering that: different searches are requested for the same information; they are coming from different origins; and they have at least one common node c in the path, preferably before reaching the destination, then, the replication would be in c, thus reducing the size of the path, and consequently improving efficiency.

In this context, to understand the proposed concept of convergence, consider several search requests, made by different nodes of the network, looking for the same identifier k. Note that the paths traveled by the requests will generate a tree with node k as the root (given that it is the destination node for all the search requests). In this tree, the vertices represent the Chord nodes through which the search request was forwarded.

In the context of the tree mentioned above, consider that:

- The root of the tree corresponds to level zero;
- N_x is the node with identifier x. present at some level i;
- $sub\_tree(N_x)$ is the number of nodes on the subtree whose root is N_x;
- $descendents(i)$ is number of nodes on all levels greater than i.

In this work, we define the convergence of node N_x that is present at level i as:

$$convergence(N_x, i) = \frac{sub\_tree(N_x)}{descendents(i)} \tag{1}$$

Figure 3 shows an example of a tree whose root, at level zero, is node N12. At level $i = 1$ are nodes N10 and N11. According to the definition: *descendents*(1) = 9, corresponding to nodes N1 to N9; *sub_tree*(N10) = 6, corresponding to nodes N1 to N6 and *sub_tree*(N11) = 3, corresponding to nodes N7 to N9. The convergence of N10 will be *sub_tree*(N10) ÷ *descendents*(1) = 6 ÷ 9, and the convergence in N11 will be *sub_tree*(N11) ÷ *descendents*(1) = 3 ÷ 9. Note that the convergence at the root will be 1.

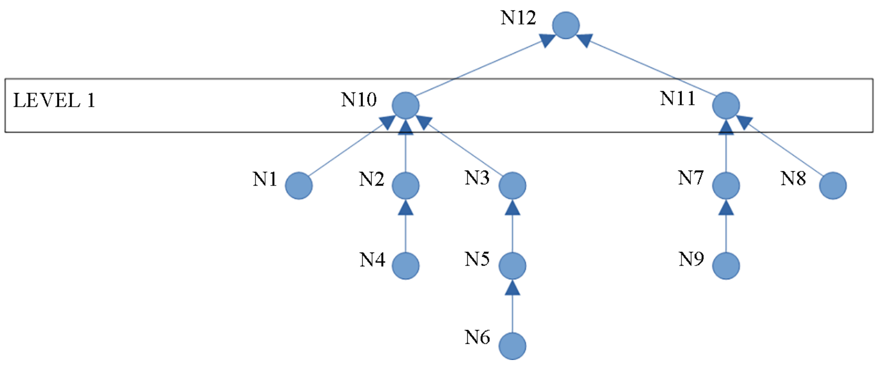


Figure 3. Convergence at level i = 1 of the tree

From the tree of paths created by the request forwarding of searches, it is possible to calculate and analyze the convergences at each level of the tree. With this, it will be possible to understand at which levels (and in which nodes of these levels) the information could be replicated to improve search efficiency.

It is important to note that, for this work, we assume the tree has static paths, meaning they do not change with dynamic behaviors, such as adding or removing nodes. Additionally, the destination node is the same for all nodes, which avoids having different destination nodes and, consequently, different trees.

5. Experimental evaluation

The experiments are carried out via simulation, considering a network size composed of n nodes. The calculated results have 95% confidence intervals within 5% of the mentioned values, for a total of 30 simulation runs. In the simulation, each Chord node makes a search request for the same identifier *k* (i.e., all searches will arrive at the same destination node), initially obtained randomly.

5.1 *Metrics*

Consider a search tree with level $i = 0$ as the root and height h (i.e., the greatest distance between the root and any node). For each level i of the tree, the following metrics were obtained:

- Conv1 corresponds to the value of the greatest convergence at level i. In other words, how much the most important node at this level contributes to convergence;
- Conv5 corresponds to the average of the five largest convergence values at level i. In other words, how much the 5 most important nodes in this level contribute to convergence;

- Conv10 corresponds to the average of the ten largest convergence values at level *i*. In other words, how much the 10 most important nodes in this level contribute to convergence.

With these metrics will be possible to understand how many searches originating at levels greater than *i* converge at 1, 5, and 10 nodes at level *i*, respectively. In any metric, a high value represents high convergence. A low value represents low convergence.

5.2 *Results*

In the graphs of the following figures, it is possible to observe the averages of the three metrics mentioned above (i.e., Conv1, Conv5, and Conv10). In each graph, the X-axis represents the level from the destination node. In other words, how many request forwards are left to reach the destination node, disregarding the level zero that only has the destination (consequently, with 100% convergence). The Y-axis represents the convergence at the level, in percentage, for each of the three metrics mentioned above.

Figure 4 shows the metrics for a small network size of 1000 nodes. In this figure, it is possible to observe that, for one extreme of the tree (i.e., level 1), the most important node, Conv1, has a convergence of 69%, the 5 most important nodes, Conv5, have a convergence of 98%, and the 10 most important nodes, Conv10, have a convergence of 100%. For the other extreme of the tree (i.e., its height or level 10), Conv1 has a convergence of 54%, and both Conv5 and Conv10 have a convergence of 100%. Approximately in the middle of the tree (i.e., level 6), which has the lowest level of convergence, Conv1 has a convergence of 15%, Conv5 has a convergence of 40%, and Conv10 has a convergence of 54%. The data was consolidated in Table 2, where the column *m* corresponds to the level with the lowest convergence and the column *h* corresponds to the height of the tree.

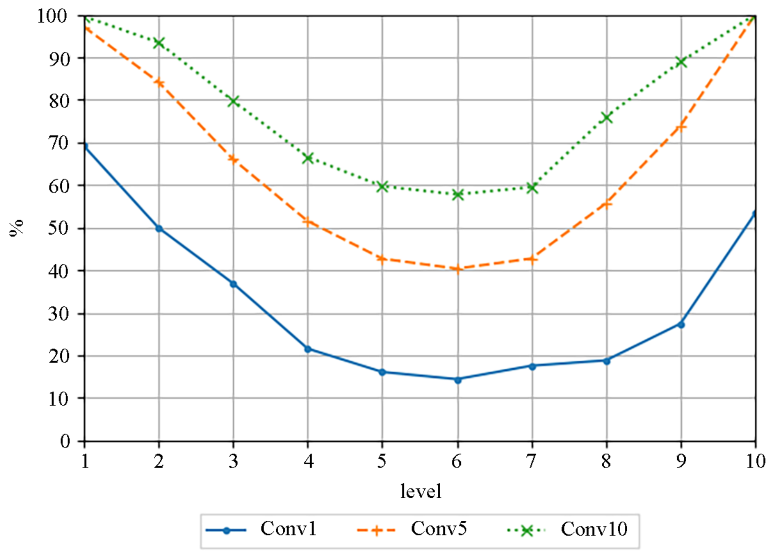


Figure 4. Convergence in a small network with 1000 nodes

Table 2. Convergence values for a network with 1000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	69%	15%	54%
Conv5	98%	40%	100%
Conv10	100%	58%	100%

Considering the difference among the curves of the three metrics, it is possible to observe that the distance between them, given a specific level, remains approximately the same (if the extreme levels are not considered—i.e., level 1 and level *h*, the height of the tree) and the distance is more noticeable from the middle of the tree.

Figures 4–10 show the tree metrics for network of sizes with 1000, 5000 (small networks), 10,000, 50,000, 100,000 (medium networks), 500,000 and 1,000,000 nodes (large networks), respectively. Tables 2–8 present the consolidation of the data obtained in the metrics for the corresponding network sizes mentioned above. It should be noted that the distance between the curves has the same characteristic mentioned for the 1000 node network, but with smaller values of convergence.

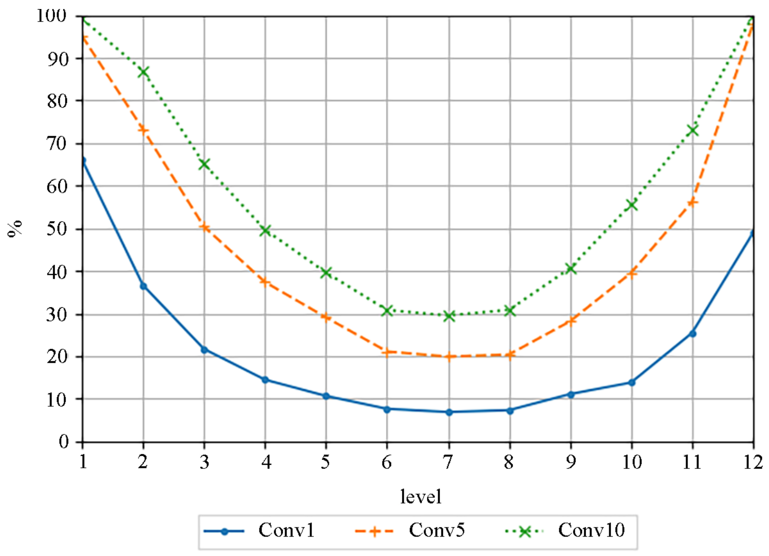


Figure 5. Convergence in a small network with 5000 nodes

Table 3. Convergence values for a network with 5000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	65%	8%	50%
Conv5	98%	20%	100%
Conv10	100%	30%	100%

Table 4. Convergence values for a network with 10,000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	63%	4%	50%
Conv5	98%	13%	100%
Conv10	100%	20%	100%

Table 5. Convergence values for a network with 50,000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	65%	3%	46%
Conv5	98%	8%	97%
Conv10	100%	10%	100%

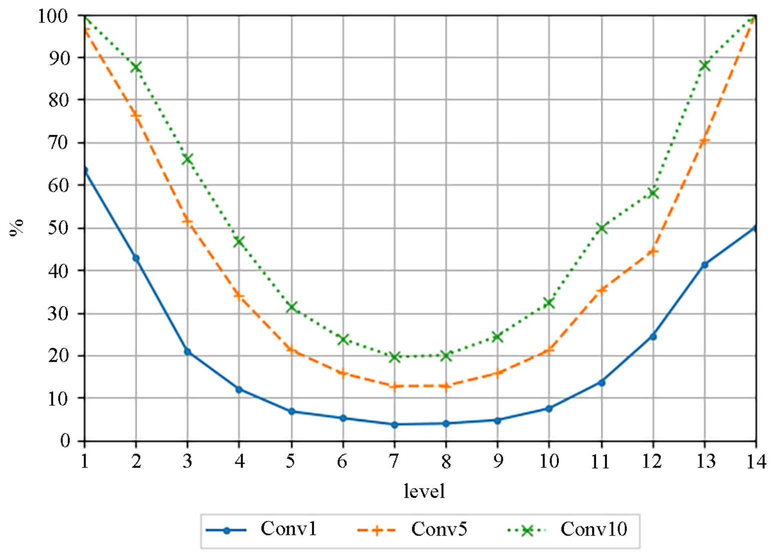


Figure 6. Convergence in a medium network with 10,000 nodes

Table 6. Convergence values for a network with 100,000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	67%	1%	24%
Conv5	98%	2%	95%
Conv10	100%	4%	100%

Table 7. Convergence values for a network with 500,000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	67%	1%	45%
Conv5	98%	2%	95%
Conv10	100%	3%	100%

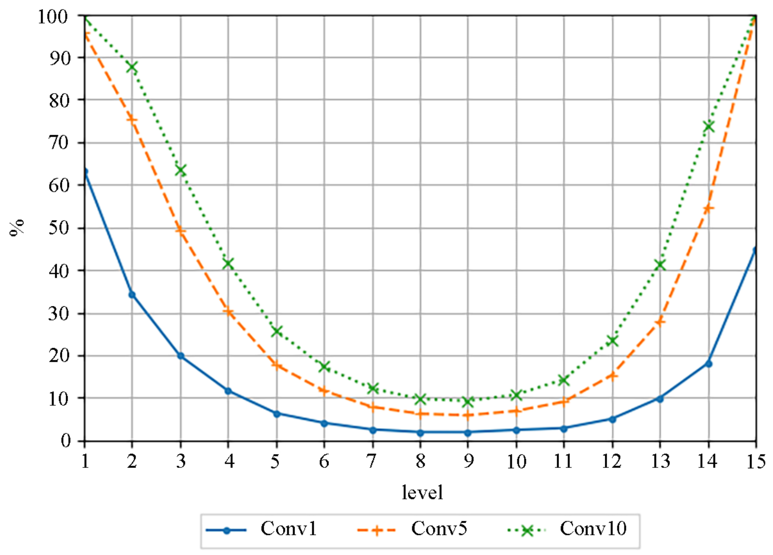


Figure 7. Convergence in a medium network with 50,000 nodes

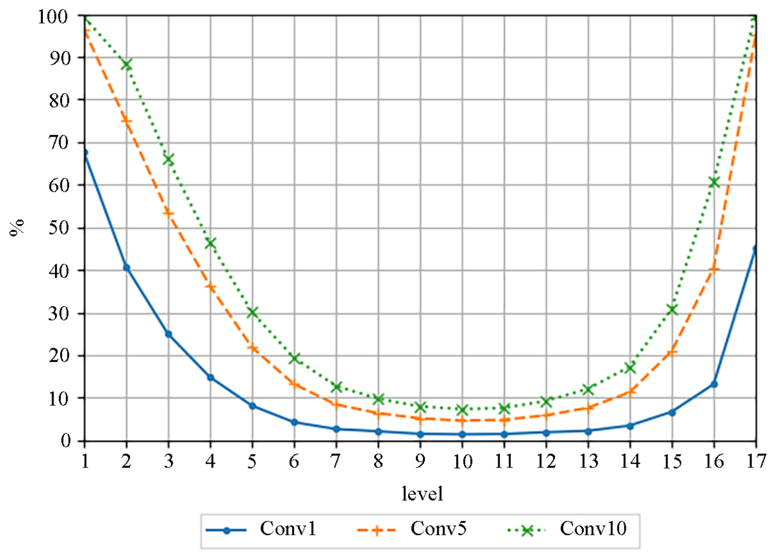


Figure 8. Convergence in a medium network with 100,000 nodes

5.3 *Discussion*

The figures presented allow for the visualization of convergence values for networks of different sizes. This aids in determining the information replication according to the threshold necessary to reduce the search path and, consequently, improve its efficiency. For example, in a small network of 5000 nodes (Figure 5), 20% of the requests could be reduced if the information is replicated in levels 1 to 3 (near the root) or in levels 11 and 12 (near the leaf). A similar scenario applies to a medium network of 50,000 nodes (Figure 7). However, in contrast, for a large network of 500,000 nodes (Figure 9), to reach the same threshold of 20% the replication must be done just in levels 1 to 3.

Furthermore, from the figures shown, it is also possible to observe that the values of the metrics form similar curves (in a U-shape), although with different values. Each curve can be separated into three zones as the tree levels increase: decreasing zone, approximately horizontal zone, and increasing zone. Table 9 consolidates the mapping between the zones and the tree levels. For example, in a network with 1000 nodes, the decreasing zone corresponds to levels 1–3, the approximately horizontal zone corresponds to levels 4–6, and the increasing zone corresponds to levels 7–10.

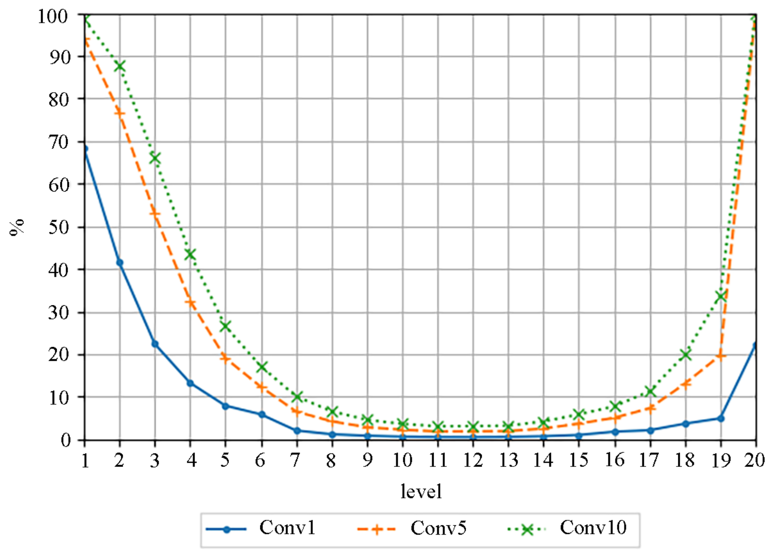


Figure 9. Convergence in a large network with 500,000 nodes

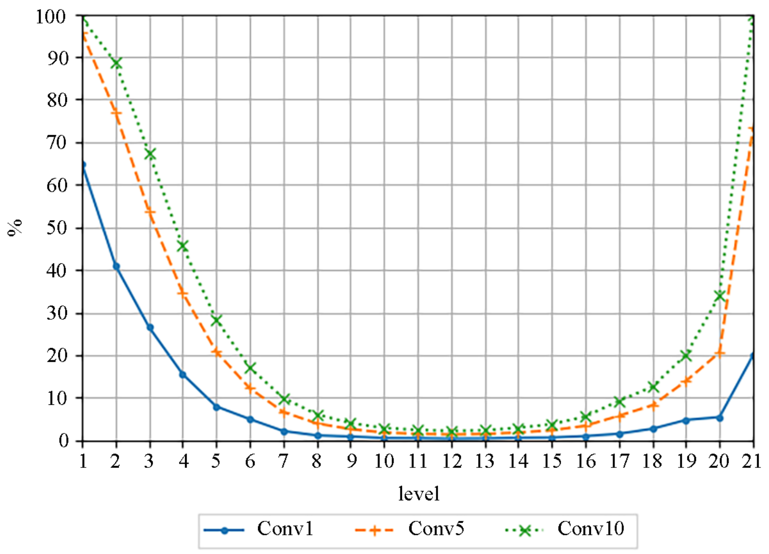


Figure 10. Convergence in a large network with 1,000,000 nodes

Table 8. Convergence values for a network with 1,000,000 nodes

Metrics	Level 1	Level *m*	Level *h*
Conv1	65%	1%	20%
Conv5	98%	2%	73%
Conv10	100%	3%	100%

Table 9. Mapping zones to tree levels

Nodes	Size	Decreasing Zone	Horizontal Zone	Increasing Zone
1000	Small Levels	1–3	4–6	7–10
5000	Small Levels	1–4	5–8	9–12
10,000	Medium Levels	1–5	6–9	10–14
50,000	Medium Levels	1–6	7–11	12–15
100,000	Medium Levels	1–7	8–12	13–17
500,000	Large Levels	1–7	8–15	16–20
1,000,000	Large Levels	1–7	8–16	17–21

The emergence of these three zones occurs due to the tree format generated by the search paths. In the tree, the root (i.e., the destination node) has limited amount of nodes in its subsequent level, which gives higher convergence values in the decreasing zone. Note that this behavior is similar closer to leaf nodes, representing the increasing zone, which also have a limited amount of nodes in their subsequent level. The horizontal zone occurs because the node is near the middle of the tree, allowing several paths both to the destination and from the leaves, resulting in lower convergence values in this zone. In the following, we analyze the convergence in these three zones which manifest independently of the network size.

Convergence in the decreasing zone. In the decreasing zone, it is possible to observe a high convergence, e.g., levels 1–3 of Figure 4 or levels 1–7 of Figure 10. This occurs because the search requests that converge in the decreasing zone also converge with other search requests in the horizontal zone, which, in turn, also converge with other search requests in the increasing zone. In this context, as there are few nodes at the levels of this zone (given that they are very close to the root), the subtrees of these nodes are large compared to the total number of nodes at the higher levels, raising the value of convergence for each node at the levels of this zone.

According to the values presented and following the results of previous research, it is possible to replicate an information in this zone to improve efficiency by reducing the path to the node responsible for the information. Note that, differently from previous works that just mention a general convergence statement, now it is possible to know the exact convergence at each level of the tree.

Convergence in the increasing zone. In the increasing zone, it is also possible to observe a high convergence, e.g., levels 7–10 of Figure 4 or levels 17–21 of Figure 10. Unlike the rationale mentioned above, this occurs because there are few nodes in this zone (given that they are very close to the leaves of the tree). In this sense, the few nodes at the levels of this zone have few descendants, consequently, increasing the value of the convergence of these nodes.

According to the values presented, in this zone, it is also possible to replicate information to improve search efficiency. It should be noted that the value of this zone is less than that of the decreasing zone for large networks.

Convergence in the horizontal zone. In the approximately horizontal zone, it is possible to observe a low convergence, e.g., levels 4–6 of Figure 4 or levels 8–16 of Figure 10. This happens because in this zone there are many nodes at a certain level, each with many descendants. As the result, the subtrees are small compared to the total number of nodes at higher levels, which decreases the convergence value for each node in this zone.

According to the values presented, in this zone, the information could be replicated in small networks, but should not be replicated in medium and large networks, as it would be necessary to replicate in many nodes to improve search efficiency

Convergence by the network size. According to the figures, it is possible to notice that the convergence in the different zones decreases when the number of nodes that form the network increases. This occurs because, as the network size increases, so does the number of nodes at each level, which increases the denominator in the convergence formula, consequently decreasing the convergence value. However, for medium and large networks (e.g., 500,000 and 1,000,000 nodes), the difference in values is not as pronounced in the decreasing and horizontal zones as in small networks (e.g., 1000 and 5000 nodes).

6. Conclusions and future work

The DHT Chord is a distributed structure widely used in various contexts to perform efficient searches for information. To improve search efficiency, researchers often utilize replication in the nodes closer to the one responsible for the information yet they do not analyze the required proximity of these nodes. To measure this proximity, this article elaborated a convergence formula for the DHT Chord and analyzed its values for different network sizes. The results revealed three distinct zones: decreasing, horizontal, and increasing. The decreasing zone is characterized by high convergence values, as nodes are closer to the destination node and there are few paths to reach it. The horizontal zone is characterized by low convergence values, attributed to the multitude of paths leading to the destination node. The increasing zone is characterized by high convergence values, as there are few path originating from the origin node. Through the simulations, it was observed that in the first and last zones, regardless of the network size, it is possible to replicate the information, and in small networks, it is possible to replicate in the horizontal zone. In addition, it was possible to verify that convergence decreases as the network size increases, but the difference is negligible when comparing just small, medium, or large sizes.

As future work and considering the limitations of this article, we point out the insertion of failures in the paths and searches, as well as exploring node mobility. Finally, there are other DHTs, such as Pastry and Kademlia that, like Chord, also create search paths when requesting specific information. In this context, we conjecture that the convergence formula may be applied and that the three zones will be formed, too.

Conflict of interest

There is no conflict of interest for this study.

References

[1] A. Bhattacharya, Z. Yang, and S. Zhang, "Temporal-DHT and Its Application in P2P-VoD Systems," in *Proc. 2010 IEEE Int. Symp. Multimedia*, Dec. 2010, pp. 81–88, https://doi.org/10.1109/ISM.2010.21.

[2] Y. Hassanzadeh-Nazarabadi, A. Kupçu, and O. Ozkasap, "Decentralized Utility- and Locality-Aware Replication for Heterogeneous DHT-Based P2P Cloud Storage Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 5, pp. 1183–1193, May 2020, https://doi.org/10.1109/TPDS.2019.2960018.

[3] V. Rocha and A. A. Franco Brandão, "A scalable multiagent architecture for monitoring IoT devices," *J. Netw. Comput. Appl.*, vol. 139, pp. 1–14, Aug. 2019, https://doi.org/10.1016/j.jnca.2019.04.017.

[4] Y. Hassanzadeh-Nazarabadi, A. Kupçu, and O. Ozkasap, "LightChain: Scalable DHT-Based Blockchain," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2582–2593, Oct. 2021, https://doi.org/10.1109/TPDS.2021.3071176.

[5] Z. Nie, J. Li, F. Duan, and Y. Lu, "A collaborative ledger storing model for lightweight blockchains based on Chord Ring," *J. Supercomputing*, vol. 80, no. 4, pp. 5593–5615, Mar. 2024.

[6] H. Zhong and H. Dai, "A Cooperative Distributed File System Based on DHT Algorithm," in *Proc. 2022 Int. Conf. Wireless Commun., Networking Appl. (WCNA 2022)*, Singapore, 2023, pp. 390–397, Springer Nature Singapore, ISBN 978-981-99-3951-0.

[7] M. Ghaleb, M. Mustafa, and F. Azzedin, "Towards Scalable and Efficient Architecture for Modeling Trust in IoT Environments," *Sensors*, vol. 21, no. 9, 2986, 2021, https://doi.org/10.3390/s21092986.

[8] Y. Hassanzadeh-Nazarabadi, S. Taheri-Boshrooyeh, and O. Ozkasap, "DHT-based Edge and Fog Computing Systems: Infrastructures and Applications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2022, pp. 1–6, https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798218.

[9] S. Zahid, K. Ullah, A. Waheed, S. Basar, M. Zareei, and R. R. Biswal, "Fault Tolerant DHT-Based Routing in MANET," *Sensors*, vol. 22, no. 11, pp. 1–18, Jun. 2022, https://doi.org/10.3390/s22114280.

[10] J. Li, C. Li, Z. Fang, H. Wang, and Y. Wu, "Optimal layer division for low latency in DHT-based hierarchical P2P network," *Int. J. Netw. Manag.*, vol. 26, no. 2, pp. 95–110, Mar. 2016, https://doi.org/10.1002/nem.1922.

[11] H. S. Nguyen, "Topology Optimization for Heterogeneous DHT-Based Multicast," in *Proc. G. Kecskemeti, Appl. Integr. Tech. Methods Distrib. Syst. Technol.*, 2019, pp. 134–155.

[12] I. Stoica, et al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proc. 2001 SIGCOMM*, Aug. 2001, pp. 149–160, https://doi.org/10.1145/383059.383071.

[13] S. Cherbal, A. Boukerram, and A. Boubetra, "RepMChord: A novel replication approach for mobile Chord with reduced traffic overhead," *Int. J. Commun. Syst.*, vol. 30, no. 14, e3285, 2017, https://doi.org/10.1002/dac.3285.

[14] H. Zhang, Y. Wen, H. Xie, N. Yu, *Distributed Hash Table: Theory, Platforms and Applications*. New York, NY, USA: Springer, 2013.

[15] P. Zave, "Reasoning about Identifier Spaces: How to Make Chord Correct," *IEEE Trans. Software Eng.*, vol. 43, no. 12, pp. 1144–1156, Oct. 2016, https://doi.org/10.1109/TSE.2017.2655056.

[16] F. Dabek, et al., "Building peer-to-peer systems with Chord, a distributed lookup service," in *Proc. Eighth Workshop on Hot Topics in Operating Systems*, 2001, pp. 81–86, https://doi.org/10.1109/HOTOS.2001.990065.

[17] A. Naghizadeh and A. Shahbahrami, "Binary search routing equivalent (BSRE): A circular design for structured P2P networks," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 4, e3012, 2017, https://doi.org/10.1002/ett.3012.

[18] D. Nguyen, N. Hoang, B. M. Nguyen, and V. Tran, "PSPChord—A Novel Fault Tolerance Approach for P2P Overlay Network," in *Proc. Smart Comput. Commun.: 3rd Int. Conf. SmartCom 2018*, Tokyo, Japan, Dec. 10–12, 2018, pp. 386–396, ISBN 978-3-030-05755-8.

[19] A. Guezzi, A. Lakas, A. Korichi, and S. Cherbal, "Peer to Peer Approach based Replica and Locality Awareness to Manage and Disseminate Data in Vehicular Ad Hoc Networks," *Int. J. Comput. Netw. Commun.*, vol. 12, pp. 65–81, Nov. 2020, https://doi.org/10.5121/ijcnc.2020.12605.

[20] S. K. Singh, C. Kumar, and P. Nath, "Resource-Cardinality Based Scheme to Reduce Resource Lookup Cost in Structured P2P Networks," *Wirel. Pers. Commun.*, vol. 125, no. 4, pp. 3351–3377, Aug. 2022, ISSN 0929-6212, https://doi.org/10.1007/s11277-022-09714-x.

[21] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, 202–215, Sep. 2001, https://doi.org/10.1145/502034.502054.

[22] K. P. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. 2003 Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Oct. 2003, pp. 381–394, https://doi.org/10.1145/863955.863998.

[23] M. Artigas, P. López, and A. Skarmeta, "On the Relationship between Caching and Routing in DHTs," in *Proc. 2007 IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.—Workshops*, Dec. 2007, pp. 415–418, ISBN 0-7695-3028-1, https://doi.org/10.1109/WI-I.